НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК_____

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
"___"_____2019р.

Магістерська дисертація

зі спеціальності – 121 Інженерія програмного забезпечення
за спеціалізацією – Програмне забезпечення розподілених систем

на тему
Інструментальні засоби розробки децентралізованих реєстрів на основі технології блокчейн

Виконав: студент _6_курсу, групи__*ТВ-з81*_
*Малюк Максим Олександрович*__ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2019

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти другий, магістерський

Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

_____ _____
(прізвище, ініціали) (підпис)

«____» _____ 2019 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри
Коваль О.В._____
(прізвище, ініціали) (підпис)

«____» _____ 2019р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Малюку Максиму Олександровичу
(прізвище, ім’я, по батькові)

1. Тема дисертації «Інструментальні засоби розробки децентралізованих реєстрів на основі технології блокчейн»
Науковий керівник Тарнавський Юрій Адамович, к.ф.н.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “ 4 ” листопада 2019 року № 3813-с
2. Строк подання студентом дисертації 9 грудня 2019
3. Об’єкт дослідження технології блокчейн для розробки децентралізованих реєстрів
4. Предмет дослідження інструментальні засоби розробки децентралізованих реєстрів правочинів
5. Перелік питань, які потрібно розробити
розробити засоби розгортання мережі децентралізованого реєстру,
розробити смарт-контракт, що використовується для створення,
відтворення та підписання договорів,
розробити API для взаємодії з децентралізованим реєстром,
створити концептуальну систему децентралізованого реєстру з
використанням розроблених інструментальних засобів

6. Перелік ілюстративного матеріалу *Назви слайдів: «Мета роботи», «Завдання магістерської роботи», «Існуючі інформаційні рішення», «Існуючі моделі*

побудови інформаційних систем», «Технологічні засоби розробки», «Технологія в основі системи – Blockchain», «Різниця між публічним та приватним блокчейном», «Архітектура системи: схема руху даних», «Мережа», «Засоби розробки: API», «Веб-інтерфейс», «Рекомендації користувачам-розробникам», «Рекомендації користувачам-клієнтам», «Висновки»

7. Перелік публікацій

Малюк М. О. АПОСТИЛІЗАЦІЯ ДОКУМЕНТІВ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ БЛОКЧЕЙН / М. О. Малюк // «СУЧАСНА НАУКА: ПРОБЛЕМИ І ПЕРСПЕКТИВИ» МАТЕРІАЛИ V МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ 29-30 ЖОВТНЯ 2019 РОКУ (частина ІІ) – Київ: МЦНІД, 2019. – (Міжнародний центр науки і досліджень). – (УДК 005). – С. 46–47.

7. Дата видачі завдання «____»_____201___р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Вивчення та аналіз задачі	1.04.2019 – 10.05.2019	
2	Розробка архітектури та загальної структури системи	11.05.2019 – 15.05.2019	
3	Розробка структур окремих підсистем	16.05.2019- 10.06.2019	
4	Програмна реалізація системи	11.06.2019 – 15.09.2019	
5	Написання та оформлення пояснювальної записки	16.09.2019 – 5.11.2019	
6	Передзахист	22.11.19	
7	Захист		

Студент _____ Малюк М.О
(підпис) (прізвище та ініціали)

Науковий керівник _____ Тарнавський Ю.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розроблений програмний продукт – інструментальні засоби розробки децентралізованих реєстрів договорів – призначений для розробки децентралізованих реєстрів договорів, що дозволяють зберігати інформацію та тексти договорів на приватному блокчейні, підписувати та відтворювати їх. Системи, реалізовані з допомогою програмного продукту, можуть взаємодіяти з реєстром напряму через командну консоль, через HTTP запити або спеціальний веб-інтерфейс. При користуванні даним продуктом користувачі-розробники мають змогу створювати реєстри, в яких кожен факт збереження, зміни, видалення або відтворення інформації записується в блокчейн і повідомляється кожному учаснику мережі, що забезпечує відкритість та доступність даних для учасників без розкриття конфіденційної інформації.

Пояснювальна записка до роботи має обсяг 95 сторінок та включає 25 ілюстрацій і 3 додатки.

Даний програмний продукт може стати концептуальним прототипом для подальших проектів у сфері Legal Tech в контексті впровадження децентралізованих реєстрів документів у правовий апарат держави.

Ключові слова: blockchain, legal-tech, smart contracts, право, розподілені сховища даних, криптографія, веб-розробка, електронний документообіг.

АННОТАЦИЯ

Разработанный программный продукт - инструментальные средства разработки децентрализованных реестров договоров - предназначен для разработки децентрализованных реестров договоров, позволяющие хранить информацию и тексты договоров на частном блокчейни, подписывать и воспроизводить их. Системы, реализованные с помощью программного продукта, могут взаимодействовать с реестром напрямую через командную консоль, через HTTP запросы или специальный веб-интерфейс. При пользовании данным продуктом пользователи-разработчики могут создавать реестры, в которых каждый факт сохранения, изменения, удаления или воспроизведение информации

записывается в блокчейн и сообщается каждому участнику сети, обеспечивает открытость и доступность данных для участников без раскрытия конфиденциальной информации.

Пояснительная записка к работе имеет объем 95 страниц и включает 25 иллюстраций и 3 приложения.

Данный программный продукт может стать концептуальным прототипом для последующих проектов в сфере Legal Tech в контексте внедрения децентрализованных реестров документов в правовой аппарат государства.

Ключевые слова: blockchain, legal-tech, smart contracts, право, распределены хранилища данных, криптография, веб-разработка, электронный документооборот.

SUMMARY

Developed software – instrumental tools for development of a decentralized treaty ledgers – is intended to be used for development of decentralized treaty ledgers, which allow users to store information and texts of contracts on a private permissioned blockchains, sign and browse stored documents. Systems, developed with use of the product, are able to interact with the ledger directly via console, using HTTP requests or web-interface. The product also enables its users-developers to easily create ledgers in which every act of data creation, modification, removal or browsing are recorded in blockchain and broadcasted to every network peer, thus ensuring the transparency and accessibility of data for all members of the network without distributing any confidential data

The explanatory note to the work has a size of 95 pages and includes 25 illustrations and 3 attachments.

This software product can become a conceptual prototype for further projects in the field of Legal Tech, namely for the introduction of decentralized public data records in the state legal apparatus.

Keywords: blockchain, legal-tech, smart contracts, law, distributed data storage, cryptography, web development, electronic document flow.

ЗМІСТ

Вступ

Протягом часу існування людської цивілізації довіра між членами суспільства зарекомендувала себе як один із найважливіших елементів соціального, політичного, економічного чи будь-якого іншого типу відносин. Довіра дозволяє суб'єктам співпрацювати, планувати будь-яку організовану роботу з надією на майбутню вигоду. Ця проста річ була одним із визначальних факторів, через які людство досягло стійкого соціального, наукового та економічного розвитку - різні люди та організації мали можливість працювати разом і довіряти один одному в умовах відсутності повної взаємної довіри та неможливості гарантування добросовісності іншої сторони. З часом засоби забезпечення довіри були винайдені та запроваджені, а найбільш ефективні отримували поширення та популярність. Один із таких інструментів забезпечення довіри та організації кооперації є договір - юридично зобов'язуюча угода між кількома сторонами щодо виконання або утримання від виконання певних дій, що має юридичну силу. Сьогодні практично жодна економічна діяльність не здійснюється без укладання та виконання контрактів.

Історично контракти укладались в усній формі, у формі дії або як прояв наміру сторін[1], пізніше отримала поширення письмова форма, а після значного розвитку правових систем - нотаріально завірена форма, хоча інші форми також дозволяються в більшості випадків (за винятками відповідно до національних / міжнародних нормотворчих документів). Проте з останніми досягненнями у галузі інформаційних технологій та комунікацій стало можливим використання нових форм договорів.

Поява нових форм договорів обумовлена як новими можливостями інформаційного суспільства, так і застарілими недоліками традиційної договірної системи. Сучасне суспільство стає дедалі більше цифровим, і те саме робить його економіка.

Комерційні відносини, процеси, переговори та інші елементи також переміщуються в цифрову форму через переваги, які надає використання

електронних систем у цих сферах - швидкість обміну даними, незалежність від кордонів та відстаней, точність, менший ризик помилок, втрати даних, дешевизна використання тощо.

Зокрема, була запроваджена ідея електронного контракту як нової форми контракту. Вона забезпечує ті ж вищезгадані переваги у порівнянні з традиційними формами договорів, хоча через відносну молодість ідеї та технології вона створює певні нові проблеми. Основною перешкодою для комерційного використання нової форми контрактів в бізнесі було відсутність довіри користувачів до юридичної сили та обов'язковості електронних контрактів. Іншою перешкодою для подальшого розповсюдження електронних контрактів була відсутність довіри до безпеки електронних систем в цілому [2].

Перша проблема виходить за межі предметної області цієї роботи, в той час як друга заслуговує на увагу в контексті розвитку електронних систем.

Наразі більшість електронних документів, що мають юридичну силу, зберігаються в централізованих реєстрах правочинів, що належать державним або недержавним організаціям. Зберігання важливих юридичних актів у централізованих репозиторіях показало себе доволі зручним способом збереження, проте у даної моделі архітектури реєстрів є певні недоліки.

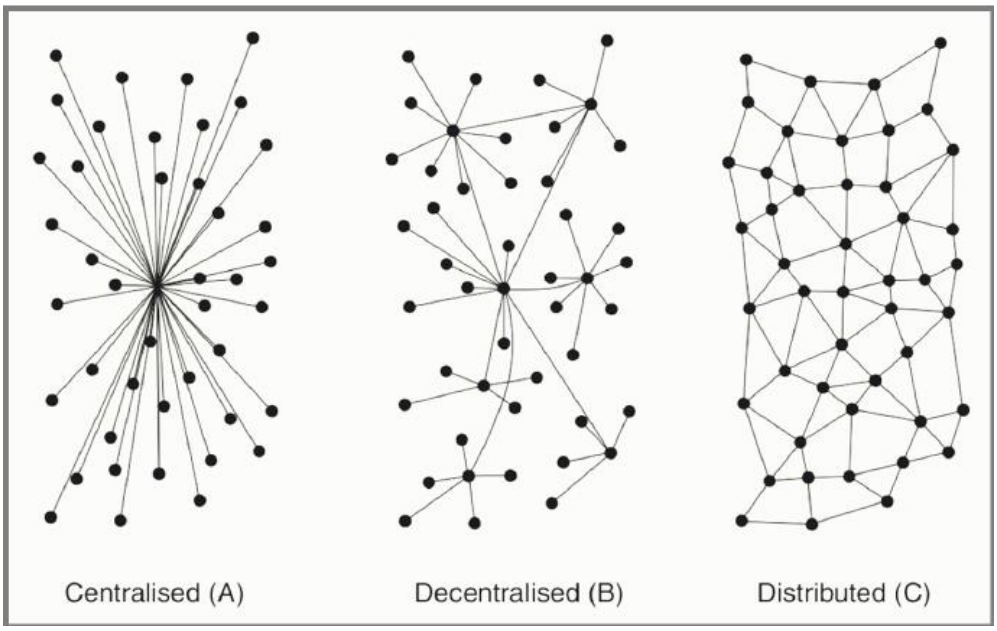

Рисунок 1. Централізований(А), Децентралізований(Б) та Розподілений(С) дизайни побудови інформаційних систем

Централізований(Рисунок 1(А)) дизайн найбільш продуктивний, коли система даних складається з невеликої кількості об'єктів і процедур. Централізований дизайн типовий для відносно малих баз даних, і може бути успішно опрацьовуватись одним адміністратором бази даних або невеликою командою розробників. Щоправда, простота побудови та обслуговування централізованих баз даних супроводжується потенційною нестабільністю та незахищеністю даних від несанкціонованого втручання. Так, якщо база даних правочинів зберігається на одному сервері, то у системи може виникнути проблема масштабовуваності коли навантаження на сервер при обробці великого масиву даних спричиняє проблеми роботі всієї системи.

Розподілений(Рисунок 1(Б)) дизайн вирішує проблему стабільності системи в контексті того, що окремі частинки системи розташовуються на різних серверах, тобто несправність одного сервера не зупинить роботу всієї системи. Однак у цього дизайну залишається основна проблема безпеки - дуже важко попередити несанкціонований доступ та зміну даних адміністратором з достатнім рівнем доступу. Всі ці проблеми спричинили необхідність пошуку альтернативи централізованим і розподіленим системам у формі децентралізованих систем.

Децентралізовані системи(Рисунок 1(С)) відрізняються тим, що система включає велику кількість вузлів, що можуть повноцінно працювати в автономному режимі і навіть повне відключення від мережі певної кількості вузлів не зупиняє її роботу. Децентралізовані системи значно складніше налаштувати і розробити, але вони природньо вирішують проблеми стабільності та (за спеціальної архітектури) несанкціонованого доступу до данних.

В умовах України питання збереження важливих документів є надзвичайно актуальною. Наприклад, щороку приватні та державні нотаріуси в Україні зобов’язані витрачати велику кількість часу на звітування про збережність документів, що долучались до процедур нотаріального затвердження документів і зберігаються в нотаріальних архівах. Нотаріат в Україні є надзвичайно важливою системою державних органів, і наразі представники держави зазначали про зацікавленість, наприклад, Міністерства Юстиції, щодо вдосконалення системи

нотаріату України, зокрема в аспекті механізму збереження нотаріально-затверджених документів у реєстрах приватних та державних нотаріусів. Варто уваги те, що одним з шляхів вдосконалення цього механізму розглядається розробка реєстру документів на технології блокчейн.

Актуальність роботи: виникнення і поширення електронних контрактів вимагає винайдення нових способів її безпечного та ефективного зберігання, поєднаного із забезпеченням її цілісності та незмінності.

Мета роботи: розробити інструментальні засоби для безпечного створення та збереження юридичних документів з використанням сучасних технологій децентралізованого зберігання даних.

Завдання роботи:

- розробити засоби розгортання мережі децентралізованого реєстру
- розробити смарт-контракт, що використовується для створення, відтворення та підписання договорів;
- розробити API для взаємодії з децентралізованим реєстром
- створити концептуальну систему децентралізованого реєстру з використанням розроблених інструментальних засобів

Вимоги до програмного продукту:

- стабільність роботи навіть при високому навантаженні системи
- неможливість прихованої зміни критичних даних
- ефективність та надійність зберігання даних
- можливість збереження конфіденційності даних без порушення принципів публічності та відкритості для уповноважених учасників
- дешевизна зберігання даних

1. Постановка задачі розробки децентралізованого реєстру правочинів

Задачі: дослідити існуючі технології на предмет можливості усунення окремих проблем централізованих інформаційних систем(відносна небезпечність, потенційна нестабільність у разі високих навантажень) та розробити на основі знайдених засобів продукт, що надаватиме інфраструктуру для створення, підписання та безпечного збереження документів.

Рішення, що відповідає даним критеріям, може стати концептуальним прототипом для подальших проектів у сфері Legal Tech як у приватному, так і державному секторі в контексті розвитку ідеї впровадження децентралізованих реєстрів даних.

Продукт включає в себе налаштовану мережу блокчейн, засоби зберігання даних в мережі та API для взаємодії користувачів з мережею.

Доступ до мережі обмежений та надаватись лише адміністратором. Користувачі, що отримали доступ до мережі, повинні мати власну цифрову особистість(криптографічний сертифікат) для авторизації операцій в мережі.

Користувачі мають права доступу до даних мережі залежно від їх позиції і ролі в Організаціях, які також є об’єктами мережі. Адміністратор Організації має можливість змінювати права доступу користувача до даних, що зберігаються в

мережі. Адміністратор Мережі має можливість змінювати права Організацій на доступ до каналів зв’язку мережі та даних, що в ній зберігаються.

Будь-які зміни в стан блокчейну є доступні до перегляду всім уповноваженим Організаціям та їх уповноваженим Користувачам. Мережа може постійно слідкувати та забезпечувати те, щоб на всіх вузлах мережі копії мережі були синхронізовані та підтримувався консенсус щодо стану блокчейну на теперішній момент.

Уповноважені користувачі можуть завантажувати, зберігати та змінювати документи в мережі через API, що є частиною програмного продукту. API дозволяє авторизованим користувачам підключатись до мережі, здійснювати створення договорів, переглядати збережені договори та їх зміст. Також API дозволяє підписувати договори де користувач зазначений як сторона договору.

Таким чином, програмне забезпечення має містити функціонал, що дозволить:

- створювати правочини у форматі, який дозволить в подальшому їх легко обробляти самій системі;
- підтверджувати факт згоди сторін договору по умовах відповідного договору;
- підтверджувати факт підписання договору сторонами;
- забезпечувати можливість ідентифікації сторін підписаного договору та їх представників;
- гарантувати неможливість прихованої зміни тексту договору після моменту його створення в реєстрі;
- гарантувати публічність та відкритість неконфіденційних даних;
- забезпечити конфіденційність умов договору від неавторизованого доступу.

Вихідні дані до проекту:

- мережа що об’єднує користувачів;

- блокчейн що зберігає дані;
- чейнкод для збереження та взаємодії з даними блокчейну;
- API для підключення до блокчейну та виконання операцій на ньому;
- служба реєстрації та авторизації користувачів.

Система має отримувати на вхід наступну інформацію:

- дані для реєстрації користувачів;
- дані для створення договорів.

На виході з системи маємо наступну інформацію:

- сховище договорів;
- історія змін та підписань договорів;
- дані користувачів;
- ієрархія користувачів в організаціях.

Користувачі системи:

- користувач;
- адміністратор.

2. Огляд існуючих програмних задач розробки децентралізованого реєстру правочинів

Наразі немає публічної інформації щодо програмних рішень, що відповідали б поставленій задачі. Найближчим по тематиці є програмне рішення Державного реєстру нерухомості Грузії, що починаючи з 20 лютого 2017 року дозволяє зберігати та отримувати інформацію щодо об'єктів нерухомості з децентралізованого сховища даних – блокчейн. Так, на офіційній веб-сторінці Національного агентства публічного реєстру додалася функція пошуку виписки в Blockchain. Через даний функціонал споживач зможе прямо з офіційної веб-сторінки Національного агентства публічного реєстру (НАПР) перевірити дійсність отриманого витягу щодо об'єкту нерухомості через інформацію, збережену в глобальній мережі Blockchain.

Окрім офіційної веб-сторінки НАПР, громадяни можуть перевірити дійсність інформації про нерухомість напряму н спеціальних веб-сторінках мережі Blockchain: blockchain.info, blocktrail.com, btc.com і ін.

З 20 лютого 2017 року інформація, підготовлена у виписці нерухомості, автоматично відправляється з Національного агентства публічного реєстру в систему Blockchain. Так як Blockchain є децентралізованою базою даних, записи, що знаходяться в ній, зберігаються в десятках тисячах комп'ютерах в усьому світі. Blockchain забезпечує захищеність, прозорість, і доступність будь-якої транзакції. Стерти, змінити, переписати, або незаконно маніпулювати потрапили в нього даними неможливо. Грузія є однією з перших країн, яка використовує технологію Blockchain під час реєстрації нерухомості.

Таким чином, даний випадок вирішення проблеми централізованих інформаційних систем показує перспективність використання технології Blockchain для вирішення подібних проблем.

3. Обгрунтування обраних програмних засобів реалізації

Для розробки та тестування продукту використовувались наступні технології та інструменти:

3.1. Операційна система

Операці́йна систе́ма, скорочено ОС (англ. operating system, OS) — це базовий комплекс програм, що виконує управління апаратною складовою комп'ютера або віртуальної машини; забезпечує керування обчислювальним процесом і організовує взаємодію з користувачем.[3]

Для написання веб-платформи було обрано виділений сервер на ОС Ubuntu 16.04. Дана операційна система являється однією з найбільш популярних на сьогодні через відносну легкість налаштування та роботи. Також саме ця ОС є зручною для розгортання оточення, необхідного для розробки даної веб-платформи.

3.2. Мова програмування та засоби реалізації

При створенні чейнкоду для децентралізованого документосховища та API для взаємодії з блокчейном використовувалась мова програмування Javascript(NodeJS 10). Дана мова забезпечує легкість розробки серверних додатків та включає багато готових високорівневих рішень, що забезпечують швидкість та гнучкість розробки продукту.

API написана за допомогою мови програмування NodeJS та бібліотеки Express. Веб-інтерфейс написаний за допомогою HTML, CSS, Javascript.

База даних була створена за допомогою CouchDB. Для надсилання запитів на API може використовуватись будь-який HTTP клієнт(curl, Postman, тощо).

Для побудови приватного блокчейну для мережі для користувачів використовується Hyperledger Fabric.

Hyperledger Fabric це закрита блокчейн-мережа, яку починали розробляти IBM та Digital Asset, яка надає можливість створення модульної архітектури з

розподілом ролей між вузлами мережі, виконання смарт-контрактів (у Fabric вони називаються "чейн-код"), гнучким механізмом досягнення консенсусу та іншими сервісами. Fabric-мережа включає "рівноправні вузли", які виконують чейн-код, мають доступ до розміщених даних, перевіряють операції та взаємодіють з додатками. "Вузли зв'язку" забезпечують стабільну роботу блокчейну та передають інформацію про перевірені операції до інших вузлів мережі. Постачальник віддалених послуг, які виконують функцію аутентифікації учасників за ролями, оперують сертифікатами X.509.

Fabric в першу чергу створювалася для інтеграції з проектами, які потребують використання технологій розподіленого реєстру. Для роботи з цією технологією потрібно лише вміти користуватися SDK для Node.js, Java або Go.

Fabric підтримує чейн-код, написаний на Go або JavaScript (через Hyperledger Composer, або нативно з версії 1.1), а також написаний на інших мовах програмування, таких як Java (після установки необхідних модулей). Такий підхід забезпечує більшу гнучкість у порівнянні з технологіями, які використовують закриті мови для створення смарт-контрактів.

3.3. База даних CouchDB

CouchDB (Cluster Of Unreliable Commodity Hardware) — розподілена документо-орієнтована система управління базами даних класу NoSQL-систем, що не вимагає опису схеми даних. Запити до CouchDB та індексація даних можуть виконуватися згідно з парадигмою MapReduce, використовуючи для формування логіки вибірки даних мову JavaScript.

CouchDB можна розглядати як сервер веб-застосунків; для реалізації цієї ідеї в CouchDB вбудований продуктивний веб-сервер, а сирцевий код, як і дані, зберігається в тій же базі даних. Для автоматизації роботи із застосунками CouchDB використовується утиліта CouchApp.

4. Теоретичні відомості про технологію блокчейн

Як зазначено в кінці розділу 1, децентралізований реєстр правочинів, розробка якого є метою даної роботи, повинен мати набір специфічний набір характеристик. Існують різні способи вирішення поставленої задачі, однак мною було обрано варіант використати природні властивості криптографічних інструментів та автономних агентів для досягнення необхідних характеристик системи.

Ключовими технологіями, що надають продукту необхідну захищеність, стійкість та гнучкість, є Blockchain(далі - блокчейн) та використання смарт-контрактів.

4.1. Технологія «блокчейн»

Блокчейн, або ланцюжок блоків транзакцій (англ. Blockchain, Block chain від block — блок, chain — ланцюг) — розподілена база даних, в якій записи є криптографічно захищені та пов’язані, що практично унеможливлює підробку записів.

Така розподілена база даних закладена в основу криптовалюти Біткоїн (вона була описана 2008 і реалізована 2009 року), Етеріум(запущено в 2015 році), де ця технологія слугує реєстром всіх операцій в мережі.

4.1.1. Історія концепції блокчейн

Перша робота над криптографічно захищеним ланцюгом блоків була описана в 1991 році Стюартом Хабером (англ. Stuart Haber) та У. Скоттом Сторнеттою (англ. W. Scott Stornetta). Вони хотіли запровадити систему, у якій неможливо змінити часові позначки документів або відновлювати їх. У 1992 році Байєр, Хабер і Сторнетта включили дерево Меркла до проекту, що покращило ефективність, дозволяючи включати декілька документів в один блок.

Концепція першого блокчейну була розроблена людиною (або групою людей), відомою як Сатоші Накамото в 2008 році. В наступному році технологію

було реалізовано і використано як основний компонент криптовалюти Bitcoin, де він служить книгою обліку для всіх транзакцій в мережі. Завдяки використанню блокчейну, Bitcoin став першою цифровою валютою з вирішеною проблемою подвійних витрат, не вимагаючи перевірених повноважень та стала натхненням для багатьох інших подібних проектів.

Станом на серпень 2014 року розмір блокчейну Bitcoin досяг 20 Гб (гігабайт). У січні 2015 року розмір виріс до майже 30 Гб, а в період з січня 2016 року по січень 2017 року - з 50 ГБ до 100 ГБ.

Слово «block» і «chain» використовувались окремо в оригінальній роботі Сатоші Накамото, але в кінцевому підсумку вони були популяризовані як єдине слово, blockchain, до 2016 року. Термін блокчейн 2.0 відноситься до нових додатків розподіленої блокчейн бази даних, яка вперше з'явиться в 2014 році. The Economist описав одну з реалізацій цього блокчейну другого покоління як «мову програмування, яка дозволяє користувачам писати більш складні та витончені контракти, створюючи, таким чином, рахунки-фактури, які сплачують себе при доставці товару або поділяють сертифікати, які автоматично направляють їх власникам дивіденди, якщо прибуток досягти певного рівня». Очікується, що вони дозволять людям входити в світову економіку, захищати конфіденційність учасників, дозволяти людям «монетизувати свою власну інформацію» та забезпечити творцям компенсацію за їх інтелектуальну власність. Технологія блокчейн другого покоління дозволяє зберігати «стійкий цифровий ідентифікатор та особу» індивідуума та надає просунутий шлях вирішення проблеми соціальної нерівності шляхом «зміни способу розподілу багатства».

У травні 2018 року Gartner виявив, що лише 1 % IT-директорів в своїх організаціях вказують будь-який тип прийняття блокчейнів, а лише 8 % IT-директорів в короткостроковій перспективі «планували або [дивляться] на активне експериментування з блокчейном».

4.1.2. Основні елементи технології блокчейн

Важливим структурним елементом блокчейну є блок. Блок транзакцій — спеціальна структура для запису групи транзакцій в системі блокчейн. Інформацію в блоках можна швидко перевірити. Кожен блок завжди містить інформацію про попередній блок. Усі блоки можна вибудувати в один ланцюжок, який містить інформацію про всі вчинені коли-небудь операції в системі. Перший блок в ланцюжку — первинний блок (англ. genesis block) — розглядається як окремий випадок, оскільки в нього відсутній материнський блок.

Блок складається із заголовка та списку транзакцій. Заголовок блоку включає в себе свій хеш, хеш попереднього блоку, хеші транзакцій та додаткову службову інформацію. Першою транзакцією в блоці завжди вказується отримання комісії, яка стане нагородою користувачеві за створений блок[10].

Далі йдуть всі або деякі з останніх транзакцій, які ще не були записані в попередні блоки. Для транзакцій в блоці використовується деревоподібне хешування, аналогічне формуванню хеш-суми файлу в протоколі BitTorrent. Алгоритм формування блоку тісно зв’язано з алгоритмом консенсусу блокчейну, найвідомішим серед яких є Proof-of-Work(«доказ працею»).

4.1.3. Алгоритми консенсусу

Консенсус означає, що всі сторони погоджуються щодо конкретного рішення. Що стосується мережі блокчейн, члени мережі досягають консенсусу щодо вмісту блокчейну.

Блокчейн - це децентралізована система, що складається з різних суб'єктів, які діють в залежно від власних інтересів та наявної у них інформації.

Всякий раз, коли нова транзакція транслюється по мережі, вузли можуть включити цю транзакцію в копію свого реєстру або проігнорувати її. Коли більшість учасників мережі приймають рішення про прийняття певного стану, досягається консенсус.

Фундаментальною проблемою в розподілених обчисленнях і багатоагентних системах є досягнення загальної надійності системи при наявності ряду

неробочих процесів. Найчастіше для цього потрібно, щоб процеси узгодили між собою деяке значення, яке знадобиться під час обчислення.

Ці процеси описуються як консенсус.

Що відбувається, коли учасник вирішує не слідувати правилам і втрутитися в стан свого Леджера?

Що відбувається, коли таких учасників в мережі досить багато, але не більшість?

Щоб консенсусний протокол був безпечним, він повинен бути відмовостійким. [23]

Проблема візантійських генералів

Надійні обчислювальні системи повинні зберігати працездатний стан навіть при відмові одного або декількох вузлів. В рамках даної роботи будемо мати на увазі під ознакою неполадок стан, в якому вузол посилає суперечливу інформацію з різних напрямів.

Завдання візантійських генералів - в криптології завдання взаємодії декількох віддалених абонентів, які отримали накази з одного центру. Частина абонентів, включаючи центр, можуть бути ворогами. Потрібно виробити єдину стратегію дій, яка буде виграшною для лояльних абонентів.

Образно завдання може бути описана наступним чином:

Візантія. У ніч перед великим боєм, Візантійська армія містить n легіонів. Кожен з них підпорядковується своєму генералу. У всій візантійській армії є головнокомандувач, що керує генералами. Імперія знаходиться в занепаді і серед генералів, включаючи головнокомандувача, можуть бути зрадники. Протягом всієї ночі, кожен з генералів отримує від головнокомандуючого наказ про дії на ранок. Це може бути один з двох варіантів: «атакувати» або «відступати». Якщо всі чесні генерали атакують - вони здобудуть перемогу. Якщо всі відступлять - їм вдасться зберегти армію. Якщо частина атакує, а частина відступить - вони зазнають поразки. Якщо головнокомандувач - зрадник, він може дати різним генералам різні накази, отже, його накази не варто виконувати беззаперечно.

Якщо ж кожен генерал буде діяти незалежно від інших, результати битви також можуть бути незадовільними. Тому генерали потребують обмінюватись інформацією, щоб дійти угоди.

Визначення проблеми

Є n генералів. Зв'язок між ними здійснюється за допомогою надійного зв'язку (наприклад, телефон). З n генералів m є зрадниками і намагаються перешкодити угоді між лояльними генералами. Згода полягає в тому, щоб всі лояльні генерали дізналися про чисельність всіх лояльних армій і прийшли до однакових висновків (будь-яким способом) щодо стану зрадницьких армій. (Остання умова є важливою, якщо генерали на підставі отриманих даних планують виробити стратегію і необхідно, щоб всі генерали виробили однакову стратегію)

Що в підсумку?

Кожен лояльний генерал повинен отримати вектор довжини n, де і-й елемент або обов'язково містить чисельність і-ої армії (в тому випадку, якщо командир лояльний), або містить довільне число в іншому випадку. Вектора повинні бути повністю однаковими у всіх лояльних генералів.

Алгоритм рішення

Крок 1: Кожен з генералів посилає іншим повідомлення, де вказує чисельність своєї армії. Зрадники можуть вказати різні числа в різних повідомленнях, а лояльні вказують істинну кількість. Генерал g1 вказав 1 (тисяча вояків), генерал g2 - 2, генерал g3 відповідно вказав трьом іншим генералам x, y, z, генерал g4 - 4. Крок 2: Кожен з генералів обчислює свій вектор з інформації, отриманої від інших генералів. Виходить: vect1 (1,2, x, 4), vect2 (1,2, y, 4), vect3 (1,2,3,4), vect4 (1,2, z, 4). Крок 3: Генерали посилають свої вектора іншим. Генерал g3 знову посилає довільні значення. Виходять такі вектори:

g_1	g_2	g_3	g_4
(1,2,y,4)	(1,2,x,4)	(1,2,x,4)	(1,2,x,4)
(a, b, c, d)	(e, f, g, h)	(1,2,y,4)	(1,2,y,4)
(1,2,z,4)	(1,2,z,4)	(1,2,z,4)	(i, j, k, l)

Рисунок 2. Вектори візантійських генералів

Крок 4: Кожен з генералів перевіряє кожен елемент в отриманих векторах. У тому випадку, якщо одне значення збігається як мінімум в двох векторах, воно поміщається в результуючий вектор. В протилежному випадку відповідний елемент позначається як «невідомий». У підсумку всі генерали отримують один вектор (1,2, невідомий, 4). Отже, згоди досягнуто. Для n = 3 і m = 1 згоди досягнуто не буде.

Ця аналогія підходить для розподілених обчислень, оскільки окремі машини (генерали) підключаються лише через мережу (месенджери). Кожен усвідомлює лише те, що спостерігає та повідомлення, які отримує. Так само кожен компонент є можливим джерелом відмов чи зловмисної діяльності.[24]

Наразі існує безліч алгоритмів консенсусу, що використовуються в різноманітних протоколах блочейнів:

- PoW(Proof-of-Work, доказ працею)
- PoS(Proof-of-Stake, доказ ставкою)
- BFT(Byzantian-Fault-Tolerance)
- Apache Kafka
- DPoS(Delegated-Proof-of-Stake, делегований доказ ставкою)[18]
- PoC(Proof-of-Capacity, доказ зберіганням даних)[19]
- PoET(Proof-of-Elapsed-Time, доказ очікуванням)[20]
- BFT(Byzantian-Fault-Tolerance, стійкість до проблеми візантійських генералів) [21]

Proof-of-Work

Proof-of-Work, або PoW, (доказ виконання роботи) - це алгоритм досягнення консенсусу в мережі блокчейн; він використовується для підтвердження транзакцій і створення нових блоків. По успішному розв'язанні задачі оператор отримує право додати до ланцюжка новий блок транзакцій та отримати за це винагороду. Основною її метою є доведення майнером своєї добропорядності шляхом пожертви свого часу та енергії на розв'язання певної математичної задачі.

Математичною задачею є проблема, що вимагають значної обчислювальної потужності. Є різні альтернативи таких проблем:

- вираховування хеш-функції з певними характеристиками, спроба знайти вхідні дані, знаючи вихідні;
- розкладання цілого числа на множники;
- «Головоломка для екскурсанта»: якщо сервер підозрює DoS-атаку, він вимагає від клієнта обчислення хеш-функцій, іноді в певному порядку, тоді це проблема обчислення значень ланцюжка хеш-функцій.

У випадку з PoW використовується хешування. У міру зростання мережі проблеми стають все серйозніше, і алгоритми хешування вимагають все більшої обчислювальної потужності, так що складність завдання - актуальна проблема.

Від механізму консенсусу залежить точність і швидкість блокчейна: проблема не повинна бути занадто складною – адже тоді генерація блоку займе багато часу, а значить, в мережі «зависне» багато незавершених транзакцій. Якщо ж проблема вирішується дуже просто – система стає вразливою для зловживань, спаму і DoS-атак.

До рішення математичної задачі також є вимога: його правильність необхідно мати можливість легко перевірити. Якщо перевірка правильності рішення, запропонованого майнером, є складним завданням, то не всі вузли зможуть зробити перевірку рішення. В такому випадку їм доведеться довіряти іншим вузлам, що змогли провести перевірку. Така ситуація би не узгоджувалась з одним із найважливіших принципів блокчейну - повною прозорістю.

Загалом алгоритм Proof-of-Work виглядає так: Майнери компонують транзакції з мережі в блок і підтверджують транзакції. Крім того, хеш кожного блоку містить в тому числі хеш попереднього блоку, що підвищує безпеку і унеможливлює порушення порядку створених блоків. Далі майнерам потрібно знайти рішення до математичної задачі мережі. Складність завдання залежить від кількості користувачів, поточної потужності і навантаження на мережу, а також проміжку часу між попереднім вирішенням задачі(саме так і оцінюється складність завдання). Якщо майнер зумів вирішити завдання, новий блок додається до кінця загального ланцюжка блоків - в ньому розміщується черговий комплект транзакцій, і вони вважаються підтвердженими, а цей хеш цього блок буде використаний в наступному циклі формування блоків та майнингу.

PoW накладає певні обмеження на дії учасників, оскільки для вирішення завдання потрібні значні зусилля та потужності. Ефективне вирішення задачі вимагає великих обчислювальних потужностей і тривалих обчислень, тому вона можлива, але невигідна через високі високих витрат електроенергії. Для складних розрахунків потрібне спеціалізоване і дороге комп'ютерне обладнання, витрати постійно зростають ростуть і майнинг стає можливий тільки для великих груп об’єднаних майнерів. Наслідком з цього стає поступове підвищення централізації системи, оскільки це вигідно. Крім того, хоча майнери виконують роботу зі створення блоків, попутно споживаючи величезну кількість енергії, по-суті самі обчислення абсолютно марні самі по собі. Так, вони гарантують безпеку в мережі, але їх результати не можна використовувати в бізнесі або в науці.

Proof-of-Stake(POS)

Proof Of Stake(Доказ ставкою) – також протокол консенсусу в мережах блокчейн, однак працює за рахунок інших економічних механік та є більш економ-ним в контексті використання природніх ресурсів, порівняно з Proof-of-Work. Доказ ставкою використовує передумову, що ті, хто володіє більшістю монет у мережі, зацікавлені підтримувати мережу, її надійність та стабільність, адже від цього залежить ціна їх власних цифрових активів.

У системі, що використовує Proof Of Stake, використовується процес випадкового обрання учасника мережі, що сформує наступний блок. Учасники можуть також вносити «ставку» своїми цифровими активами(крипто валютою цієї мережі) для того, щоб стати валідатором (тим, хто може створювати блоки), а значить, вони заморожують свої активи на певний час. Після цього вони мають право виробляти блоки. Процес, який вирішує, хто отримує наступний блок, бере до уваги кілька факторів. Як правило самі фактори залежать від протоколу і реалізації самого блокчейна, але в цілому учасник, який має найбільший пакет акцій, має найбільший шанс отримати право створити блок. Прикладом іншого чинника, який можна врахувати, є тривалість заморозки активів.

Валідатори також винагороджуються за свою роботу. Винагорода, яку отримує валідатор за створення наступного блоку, ще раз залежить від дизайну блокчейна. Зазвичай вони або отримують всю суму або частину трансакційних зборів всіх транзакцій у створеному ними блоці, або отримують фіксовану кількість валюти(генерованих за рахунок інфляції).

Proof of Stake - це не лише набагато енергоефективніша система ніж Proof Of Work, але також маю ще одну головну відмінність. У системі Proof of Work майнер отримує винагороду за виконання в мережі обчислень, тобто вони прагнуть лише максимізувати свій прибуток без фактичного вдосконалення мережі. У системі Proof Of Stake валідатори мають набагато більший стимул насправді підтримувати мережу, оскільки вони фактично закладають власні гроші як доказ того своєї добросовісності при формуванні блока. У разі ж формування неналежного блоку, він може бути відхиленим і тоді валідатор втратить не лише потенційну винагороду, а й свою «ставку».

Delegated Proof of Stake (DPOS)

Делегований доказ ставкою (DPOS) - це дуже швидкий консенсусний механізм, найбільш відомий тим, що він впроваджується в EOS(одним з найбільших блокчейн проектів), і його часто називають цифровою демократією,

завдяки системі зваженого голосування в мережі та наявності своєї внутрішньо-мережної Конституції[22].

У делегованій системі Proof Of Stake користувачі можуть закласти свою валюту, щоб проголосувати за певну кількість делегатів. Вага їхнього голосу залежить від їхньої акції, наприклад, якщо А ставить 10 монет на делегата, а В - 1 монета на делегата, голос А важить у 10 разів важче, ніж голос В.

Але що таке делегат? Делегат - це людина або організація, яка хоче виробляти блоки в мережі(block provider). Делегати, які отримують найбільшу кількість голосів, отримують блоки та отримують винагороду за створення цих блоків. Як і у Proof Of Stake, винагорода або сплачуються з комісійних платежів, або виплачуються фіксованою кількістю монет, які створюються за рахунок інфляції. Скільки делегатів отримують за створення блоків, залежить від консенсусу блокчейна, загалом це або фіксована сума, або всі делегати вище певного рівня оплати. Головні делегати можуть бути змінені голосуванням всередині мережі.

Оскільки делегати хочуть отримати якомога більше голосів, вони постійно заохочуються виконувати цінні для мережевої громади дії, оскільки вони, ймовірно, отримують додаткові голоси за це.

Delegated Proof Of Stake також набагато ефективніший при обробці транзакцій, ніж вищеперелічені протоколи. У протоколів з таким алгоритмом консенсусу значно вища TPS(Transaction-per-Second), тобто швидкість обробки транзакцій в мережі, що є великою проблемою для протоколів з Proof-Of-Work де транзакції можуть очікувати обробки та включення в блок від 10 хвилин до кількох годин.

4.1.4. Різниця між публічним та приватним блокчейном

Публічні та приватні блокчейни мають багато спільних рис: обидва види використовують децентралізовані мережі, покладаються на протоколи консенсусу для перевірки транзакцій та надають гарантії на незмінність реєстру блокчейну.

Однак на відміну від публічних блокчейнів, де кожен може брати участь у мережі, у приватних блокчейнах доступ до мережі є обмеженим і відкритий лише для тих, хто має запрошення. Ця більш обмежена блокчейн-архітектура може бути привабливішою для компаній, які прагнуть вести бізнес, не публікуючи конфіденційних даних у відкритих джерелах.

Існують також і публічні блокчейни, особливістю яких є високий рівень приватності та конфіденційності. Такі проекти(наприклад Monero), захищають публічні дані користувача(наприклад, адресу) та історію трансакцій, але все ще працюють у загальнодоступній мережі. Приватні ж блокчейни працюють на принципово іншому рівні – вони обмежують публічність та видимість. Така характеристика не порушує принципи блокчейну(відкритість та публічність), адже дані залишаються доступними і відкритими, але лише в межах мережі корпоративних партнерів.

Іноді їх називають дозволеними блокчейнами, публічні блокчейни дозволяють будь-кому брати участь у мережі. Це включає не тільки надсилання та отримання транзакцій, а й розміщення вузла та перегляд історії транзакцій. Біткойн - найвизначніший приклад публічного блокчейна. Публічні блокчейни є відкритими для всіх, і ці характеристики зробили їх надзвичайно популярними у програмах, які передбачають інтерес необмеженої кількості людей.

Алгоритм консенсусу - це те, що дозволяє мережі працювати без централізованого органу. Чим більше вузлів приєднується до мережі, тим сильніше вона стає. Однак це також означає, що мережа за своєю суттю повільна, оскільки необхідні надійні механізми консенсусу, щоб переконатися, що вузли узгоджуються.

Публічні блокчейни також повинні мати рідну валюту для живлення мережі. Валюта слугує винагородою для майнерів та власників вузлів мережі. Деякі публічні блокчейн також підтримують токенізовані активи, які можуть представляти все, що має цінність. Ethereum, який підтримує незліченну кількість токенів ERC-20, покладається на рідну монету Ether, щоб підтримувати всю мережу та надавати цінність активам всередині мережі. Приватні блокчейни,

також відомі як дозволені блокчейни, в основному внутрішньо використовуються окремими організаціями або консорціумами, які прагнуть використовувати технологію blockchain для зниження витрат з обліку реєстрів записів. Приватні блокчейни можуть існувати виключно в одній компанії або ділитися між будь-якою кількістю вибраних організацій. Є багато причин, чому бізнес частіше надає перевагу приватним блокчейнам.

По-перше, приватні блокчейни виконують багато тих же основних функцій, що і публічні. Однак найбільша різниця полягає у тому, хто має доступ до мережі. Приватні блокчейни передбачають з обмежений доступ та участь. Як результат, особистість усіх учасників відома та довірена іншим учасникам мережі.

Це забезпечує велику перевагу для підприємств, оскільки вони можуть знати, з ким вони працюють. Навіть після приєднання до приватної мережі учасники зазвичай мають обмежений доступ до даних, якими надається спільний доступ у блокчейні. Будь-які дані можуть бути зашифровані так, що лише відповідні сторони мають ключі для перегляду будь-яких конфіденційних даних у блокчейні, тоді як усі інші мережі не можуть переглядати дані транзакцій.

Особливості приватних блокчейнів дозволяють учасникам більше довіряти один одному, оскільки тільки довірені учасники будуть запрошені приєднатися до мережі. Із встановленням довіри стає менше необхідності в таких алгоритмах консенсусу, які характерні публічним блокчейнам. Натомість консенсус може бути досягнутий шляхом голосування або багатостороннього алгоритму консенсусу, який може працювати набагато швидше і дешевше, ніж безпечний публічний блокчейн який передбачає що ніхто в мережі не довіряє одне одному.

Нарешті, приватні блокчейни не завжди покладаються на рідну валюту. Всі публічні блокчейн засновані на власній специфічній крипто валюті, а приватні блокчейни цього не потребують і замість цього можуть покладаютись на будь-які цифрові активи, які представляють фізичні блага. Система винагород за майнинг чи обробку транзакцій в таких приватних блокчейнах також необов’язкова. [16]

Сценарії використання приватних блокчейнів

Банкінг. Хоча блокчейн першопочатково ставився як альтернатива централізованому сучасному банкінгу, банки - це найбільш очевидні випадки використання для технології приватнмих блокчейнів. Кожен банк веде свою власний реєстр операцій, які проводить з іншими організаціями. Зазвичай це транзакції, що створюються клієнтами, що надсилають гроші на рахунки в інших банках. Сьогодні ці перекази між банками є досить повільними та дорогими, особливо для міжнародних переказів. Кілька банків у всьому світі можуть спільно працювати над створенням приватної мережі блокчейн.

Це дозволило б більш плавно здійснювати передачу між партнерами з використанням цифрових активів та високошвидкісну продуктивність приватних блокчейнів, одночасно захищаючи дані транзакцій від широкої громадськості. Приватний блокчейн, розроблений R3, вже вступив у пілотну фазу, приєднавшись до 100 фінансових установ з торговими платформами та провідними світовими банками, такими як Commerzbank та Standard Chartered[17].

Державні реєстри. Державні органи також можуть скористатися приватними блокчейнами, використовуючи їх для підвищення безпеки даних проти хакерів та зменшення корупції. Архітектура розподіленої книги блокових ланцюгів робить злому в систему складнішим, ніж це стосується нинішніх централізованих систем.

Хакеру потрібно було б підробити запрошення всіх відповідних сторін отримати доступ до мережі замість того, щоб взламувати лише одну централізовану систему. Крім того, хакеру знову потрібно буде знайти спосіб перегляду будь-яких зашифрованих даних, що існують у приватному блокчейн, який можна переглядати лише між відповідними сторонами.

Корупція є проблемою у багатьох урядах по всьому світу. Деякі люди послідовно зловживають коштами для особистої вигоди. Використовуючи приватний блокчейн всередині державної, можна записувати, де здійснюються платежі. Незмінність розподіленої книги гарантує, що запис про використання цих коштів не може бути змінений.

Крім того, кошти можуть розподілятися лише між тими, хто є членами приватної блокчейн-мережі. Це унеможливлює пересилання грошей комусь, крім агентств та людей, яких схвалює уряд, не перетворюючи ці гроші у фіатну валюту.

Приклади приватних блокчейнів

Деякі приватні блокчейни розробляються як бізнес-рішення компаніями, переліченими нижче:

1. Hyperledger Fabric - це приватне блокчейн-рішення з відкритим кодом для підприємств. Він розроблений Intel, IBM та 26 іншими організаціями. Тканину гіперлегерів можна адаптувати до таких галузей, як банківська справа, уряд, охорона здоров'я та засоби масової інформації. Це дуже масштабоване рішення, яке також робить його придатним для застосування в ланцюгах поставок.

2. R3 розробляє блокчейн-платформу Corda для фінансів та комерції з 2016 року. Вона існує як у відкритому коді, так і в корпоративних версіях, залежно від того, що краще відповідає потребам бізнесу. R3 працює з більш ніж 200 компаніями та регуляторами на шести континентах. Банки HSBC та ING вже здійснили оперативну фінансову операцію на блокчейні Corda ще в травні 2018 року.

3. BankChain був створений Державним банком Індії в лютому 2017 року. Він створив консорціум банків для дослідження та впровадження блокчейн в банківській галузі. В даний час існує 37 членів, включаючи таких партнерів, як Deutsche Bank та Citibank, N.A. Члени здійснюють проекти в галузі транскордонних переказів, а також автентифікації та перевірки документів, загалом 8 активних проектів у стадії розробки.

5. Опис програмної реалізації

Основою програмної реалізації даного децентралізованого реєстру правочинів є технологія блокчейн.

Програмне забезпечення включає наступні структурні елементи:

- мережа
- API додаток

5.1. Мережа Hyperledger Fabric

Блокчейн мережа - це технічна інфраструктура, яка надає доступ користувачам та адміністраторам програм до записів децентралізованого реєстру.

Мережа складається з:

- реєстрів записів (одна на канал - складається з блокчейна та бази даних стану);
- каналів;
- смарт-контракти (ака-ланцюговий код);
- вузли мережі;
- впорядковувачі;
- центри сертифікації.

5.1.1. Реєстр записів

Реєстр складається з двох пов'язаних між собою частин – блокчейну та глобального стану.

Блокчейн

Як вже було зазначено глобальний стан містить сукупність фактів, що стосуються поточного стану набору бізнес-об'єктів мережі, а блокчейн - це історичний запис фактів про те, як ці об'єкти дійшли до їх сучасних станів. Блокчейн містить дані про кожну попередню версію кожного стану книги та спосіб і час її зміни.

Блокчейн структурований як послідовний журнал взаємопов'язаних блоків, де кожен блок містить послідовність транзакцій, кожна транзакція представляє

запит або оновлення до світового стану. Послідовність блоків, а також послідовності транзакцій всередині блоків встановлюється таким компонентом мережі як служба впорядкування(Ordering service) або впорядковував(Orderer).

Розподілена база даних Blockchain формується як безперервно зростаючий ланцюжок блоків з записами про всі транзакції. Копія бази даних або її частини одночасно зберігаються на безлічі комп'ютерів та синхронізуються відповідно до формальних правил побудови ланцюжка блоків. Інформація в блоках не шифрована і доступна у відкритому вигляді, однак захищена від змін криптографічно через хеш-ланцюжок.

Найчастіше умисна зміна інформації в будь-якій з копій бази або навіть в досить великій кількості копій не буде визнана істинною, оскільки не відповідатиме правилам. Деякі зміни можуть бути прийняті, якщо будуть внесені в усі копії бази (наприклад, видалення кількох останніх блоків через помилку в їхньому формуванні).

В Hyperledger Fabric блокчейн реалізовується як файл, в той час як глобальний стан має вигляд бази даних. Це зумовлено тим що, структура даних blockchain дуже впорядкована та може бути змінена невеликим набором простих операцій. Додавання до кінця blockchain є первинною операцією, і запит даних в конкретний час є відносно рідкісною операцією.

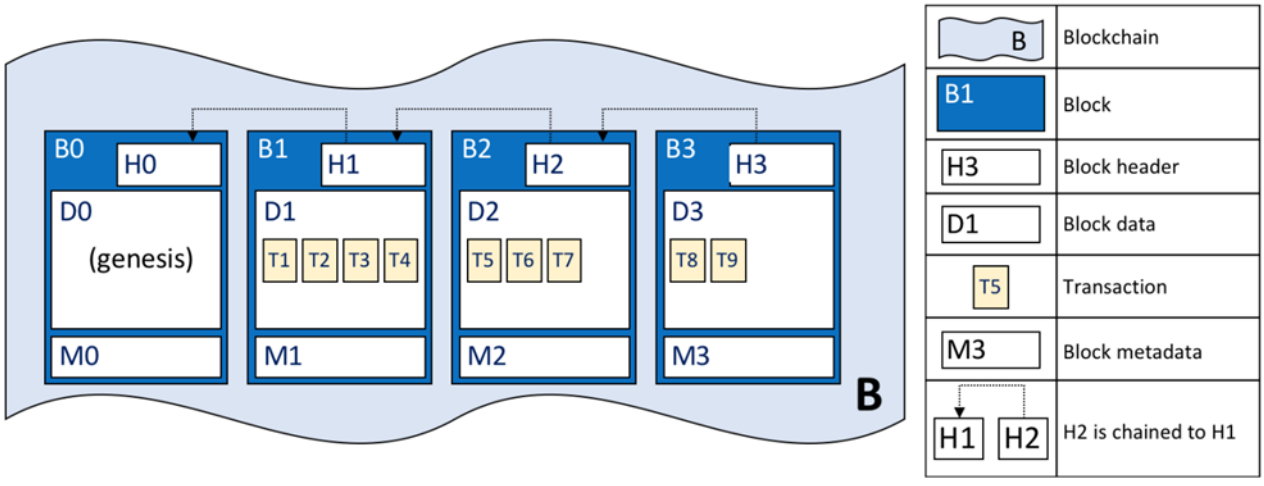

Рисунок 3. Структура блокчейну

На Рисунку 3 ми бачимо структуру блокчейну – блокчейн B включає блоки B0, B1, B2, B3. B0 є першим блоком блокчейну, або його початковим блоком, і не містить хеш попередника. Також ми бачимо що блок B2 містить дані D1, а саме транзакції T5, T6, T7. Крім того, блок B2 має заголовок H2, що містить хеш усіх транзакцій всередині D2, так само як і хеш попереднього блоку B1. Таким чином, як уже було пояснено вище, блоки нерозривно пов’язані одне з одним у послідовний ланцюг.

Кожен блок завжди містить інформацію про попередній блок(заголовок або посилання. Усі блоки можна вибудувати в один ланцюжок, який містить інформацію про всі вчинені коли-небудь операції з в мережі. Перший блок в ланцюжку — первинний блок (англ. genesis block) — розглядається як окремий випадок, оскільки в нього відсутній материнський блок. Він не містить не містить жодних транзакцій, натомість він містить транзакцію конфігурації, що містить початковий стан мережевого каналу.

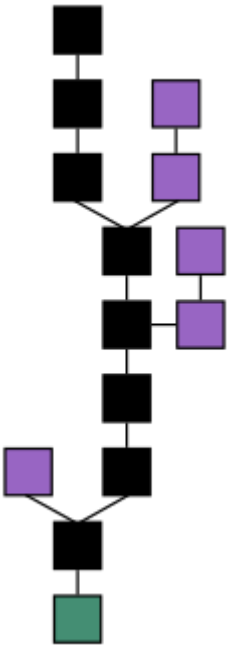

Рисунок 4. Ланцюжок блоків

Структура блоку:

- заголовок блоку. Він містить три поля, написані при створенні блоку:

o номер блоку: Ціле число, що починається з 0 (блок генезису), і збільшується на 1 для кожного нового блоку, доданого до блокчейну;

o хеш цього блоку: хеш усіх транзакцій, що містяться в поточному блоці;

o хеш попереднього блоку: копія хеша з попереднього блоку в блокчейні;

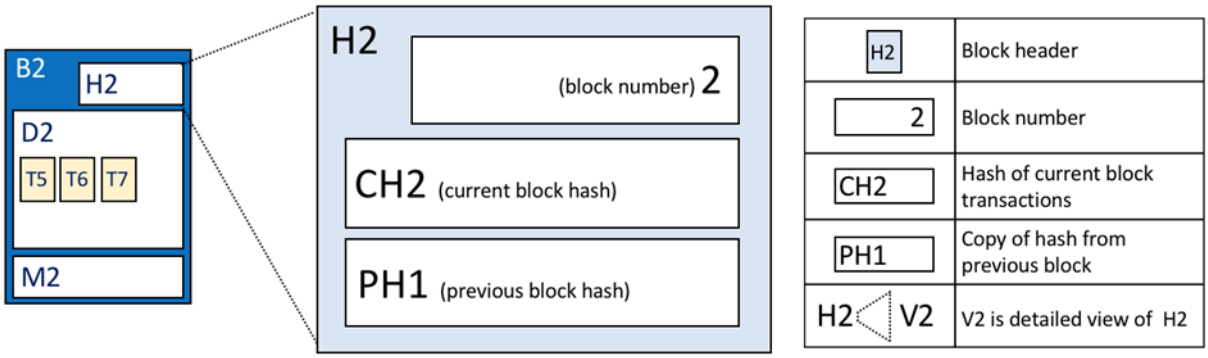

Рисунок 5. Структура блоку

На рисунку 5 заголовок H2 блоку B2 складається з блоку №2, хеша CH2 поточних даних блоку D2 та копії хеша PH1 з попереднього блоку, блоку №1.

- Список транзакцій блоку. Тут містяться транзакції, що збережені в певному порядку. Цей порядок визначається службою впорядкування мережі.
- Метадані блоку. Ця частина блоку містить час написання блоку, а також сертифікат, відкритий ключ та підпис записаного блоку.

Транзакції

Транзакція фіксує зміни до глобального стану. Давайте подивимось детальну структуру блочних даних, яка містить транзакції в блоці.

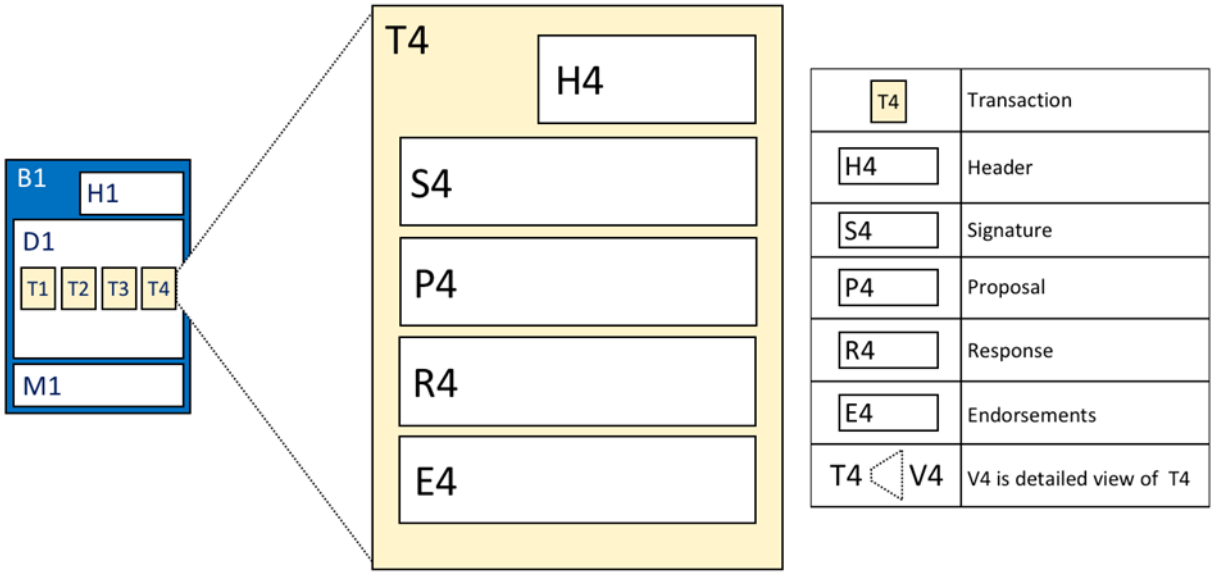

Рисунок 6. Структура транзакції

Транзакція T4 в блочних даних D1 блоку B1 складається з заголовка транзакції, H4, підпису транзакції, S4, пропозиції транзакції P4, відповіді на транзакцію, R4 та списку схвалень, E4.

Отже, транзакція має наступні поля:

- Заголовок(Header). Це поле, проілюстроване як H4, фіксує деякі основні метадані про транзакцію - наприклад, назву відповідного чейнкоду та його версію.

- Підпис(Signature). Це поле, проілюстроване як S4, містить криптографічний підпис, створений клієнтською програмою. Це поле використовується для перевірки того, що дані про транзакцію не підроблені, оскільки для їх створення потрібен приватний ключ програми.

- Пропозиція(Proposal). Це поле, проілюстроване P4, зберігає вхідні параметри, що надаються додатком до смарт-контракту, що створює запропоноване оновлення стану мережі. Коли смарт-контракт працює, ця пропозиція надає набір вхідних параметрів, які в поєднанні з поточним глобальним станом визначають новий глобальний стан.

- Відповідь(Response). Це поле, проілюстроване як R4, фіксує значення глобального стану до і після. Це результат роботи чейнкоду, і якщо транзакція

буде успішно підтверджена, вона буде застосована до реєстру для оновлення глобального стану.

- Підтвердження(Endorsements). Як показано на E4, це перелік підписаних відповідей на транзакції від кожної необхідної організації, необхідні для задоволення політики затвердження. Варто звернути увагу що хоча в транзакцію включена лише одна відповідь на транзакцію(виконана вузлом, що отримав сам запит на транзакцію від клієнта), існує кілька схвалень. Це тому, що кожне схвалення ефективно кодує особливу відповідь на транзакцію своєї організації - це означає, що не потрібно включати відповідь на транзакцію, яка не відповідає достатній підтримці, оскільки вона буде відхилена як недійсна та не впливатиме на глобальний стан.

Порядок проходження транзакції в мережі включає наступні етапи:

- Користувач ініціює транзакцію. Будується пропозиція про транзакцію. Клієнтський додаток, що використовує підтримуваний SDK (Node, Java, Python), використовує один із доступних API для створення пропозиції на транзакцію(transaction request). Пропозиція - це запит на виклик функції смарт-контракту з певними вхідними параметрами. Пропозиції бувають з наміром зчитування(для викликів функцій, що лише зчитують інформацію з блокчейну, наприклад збирає інформацію про стан договору, що було збережено в блокчейн) або оновлення реєстру(для внесення в блокчейн нових або зміну існуючих даних). Сформована пропозиція далі підписується криптографічними даними користувача.

- Вузол схвалення перевіряє підпис та виконує транзакцію. Вузли, що схвалюють, підтверджують (1), що пропозиція транзакції вірно сформована, (2) вона не була подана вже в минулому (захист від повторної атаки), (3) підпис дійсний (за допомогою MSP) та (4)) що користувач належним чином уповноважений виконувати запропоновану операцію на цьому каналі (а саме кожен одноранговий підтверджує, що суб'єкт задовольняє політику каналу "Writers").

Цей запит орієнтований на смарт-контракт S5, що встановлений вузлі P1, на якому відповідно цей смарт-контракт було встановлено. Далі транзакція повинна бути схвалена згідно Політиці схвалення операцій, яка визначає правила схвалення транзакцій в цьому каналі. Наприклад, в політиці даної мережі запрограмовано що транзакція повинна бути підтверджена ще одним вузлом організації O2, тому запит переходить до P1 та P2(вузла, що належить організації O2).

MSP - це вузловий компонент, який дозволяє вузлам перевіряти запити транзакцій, що надходять від клієнтів, і підписувати результати транзакцій (схвалення). Політика підписання встановлюється під час створення каналу та визначає, які користувачі мають право подати транзакцію на цей канал.

- Вузли, що схвалюють, приймають дані пропозиції транзакцій як параметри методу викликаного смарт-контракту. Потім смарт-контракт виконується в базі даних поточного стану для отримання результатів транзакцій, включаючи значення відповіді, набір зчитуваних даних та набір записуваних даних (тобто пари ключів / значень, що представляють дані для зчитування, створення або оновлення). На даний момент оновлення реєстру ще не проводиться. Набір цих значень разом із підписом вузла передається назад як "відповідь на пропозицію" SDK.

- Відповіді на пропозиції перевіряються. Додаток перевіряє підтвердження підписів вузлів і порівнює відповіді пропозицій, щоб визначити, чи відповіді пропозиції однакові. Якщо транзакція передбачає лише зчитання даних з реєстру, програма лише перевірить відповідь і не надсилає транзакцію до ордерера. Якщо ж транзакція передбачає зміну даних блокчейну, програма визначає, чи була виконана зазначена політика затвердження перед надсиланням (тобто, чи виконали P1 та P2 схвалення).

- Клієнт збирає підтвердження угоди. Додаток "транслює" пропозицію про транзакцію та відповіді до ордерера. Транзакція містить набори читання / запису даних, підписи вузлів схваленняв та ідентифікатор каналу. Службі замовлення не потрібно перевіряти вміст транзакції з метою виконання її операції,

вона просто отримує транзакції з усіх каналів у мережі, хронологічно їх впорядковує та формує в блоки транзакцій каналу.

- Транзакція підтверджена та здійснена. Блоки транзакцій "доставляються" всім вузлам на каналі. Операції всередині блоку перевіряються для забезпечення виконання політики затвердження та для того, щоб не було змін стану реєстру для змінних наборів зчитування, оскільки набір зчитування генерувався виконанням транзакції. Операції в блоці позначені як дійсні або недійсні.
- Реєстр оновлено. Кожен вузол додає блок до своєї копії блокчейну, і для кожної дійсної транзакції набори запису передаються в базу даних поточного стану. По каналу розповсюджується івент, що повідомляє клієнтську програму про те, що транзакція (виклик) була незмінно додана до ланцюга, а також повідомлення про те, чи була транзакція підтверджена чи визнана недійсною.

Алгоритм консенсусу блокчейну Hyperledger Fabric

Консенсус у мережі Hyperledger - це процес, коли вузли в мережі забезпечують гарантоване впорядкування транзакції та підтверджують той блок транзакцій, який потрібно здійснити в реєстрі. Консенсус повинен забезпечити наступне в мережі:

- підтверджує правильність усіх транзакцій у запропонованому блоці відповідно до політики затвердження та консенсусу;
- погоджує порядок і правильність, а отже, і результати виконання (мається на увазі угоду про глобальну державу);
- інтерфейси та залежать від рівня смарт-контракту для перевірки правильності впорядкованого набору транзакцій у блоці.

Консенсус повинен задовольняти двом властивостям, щоб гарантувати узгодження між вузлами: безпеку та життєздатність.

Безпека означає, що кожному вузлу гарантується однакова послідовність входів і приводить до одного виходу на кожному вузлі. Коли вузли отримують ідентичну серію транзакцій, на кожному вузлі відбудуться однакові зміни стану.

Алгоритм повинен поводитись ідентично одній системі вузлів, яка виконує кожну транзакцію атомним шляхом.

Життєздатність означає, що кожен несправний вузол зрештою отримає кожну подану транзакцію, якщо як тільки відновиться зв’язок з кількома вузлами, що під’єднані до мережі.

Hyperledger використовує “консенсус на основі голосування”. Він працює на основі припущення, що бізнес-мережі блокчейн(на зразок Hyperledger Fabric) учасники будуть працювати в умовах часткової довіри. Таким чином, алгоритми на основі голосування є вигідними тим, що вони забезпечують низьку кінцеву затримку обробки операцій. Коли більшість вузлів підтверджує транзакцію чи блок, консенсус існує і настає остаточність. Оскільки алгоритми на основі голосування зазвичай вимагають від вузлів для передачі повідомлень кожному з інших вузлів у мережі, чим більше вузлів існує в мережі, тим більше часу потрібно для досягнення консенсусу. Це призводить до компромісу між масштабованістю та швидкістю.

Консенсус в Hyperledger Fabric можна поділити на 3 стадії:

1. Схвалення. Процедура схвалення визначається конфігурацією мережі, а саме політикою схвалення (з n підписів), за якою учасники схвалюють транзакцію.
2. Упорядкування. На цьому етапі схвалена транзакція включається до масиву інших транзакцій в певному порядку, в якому ця та інші транзакції будуть включені до головного реєстру.
3. Перевірка. На цій стадії блок упорядкованих транзакцій перевіряється і підтверджується на критерій правильності результату.

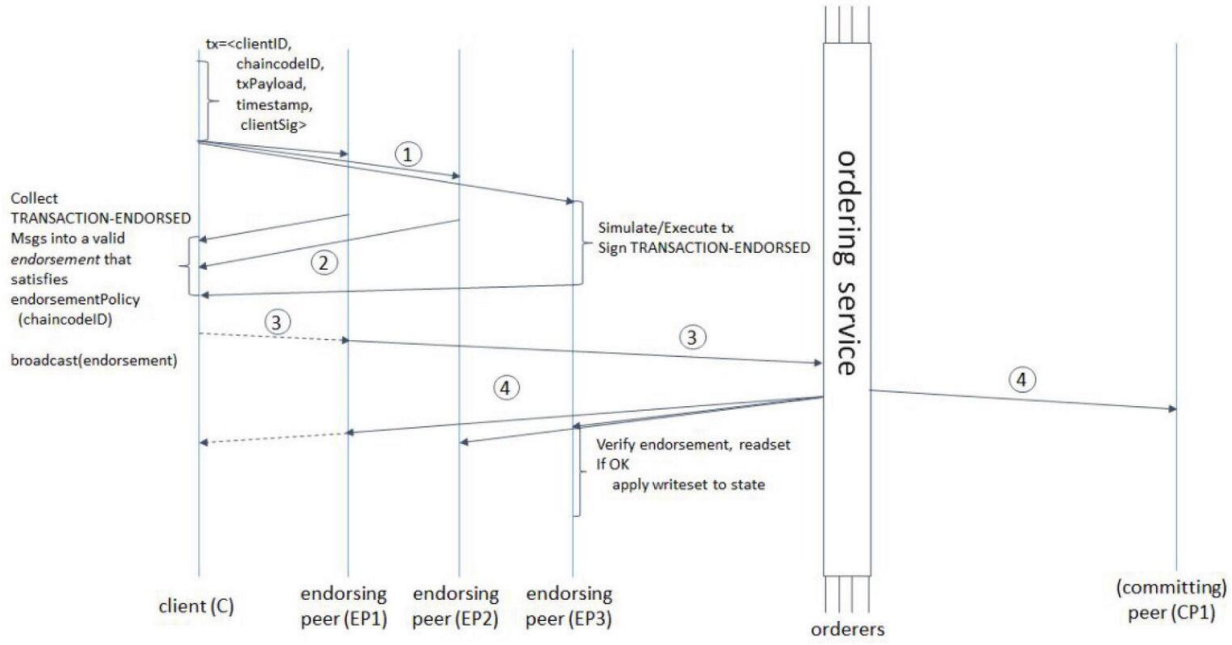

Рисунок 7. Порядок руху транзакції в Hyperledger Fabric[24]

Hyperledger Fabric дозволяє обирати різні алгоритми консенсусу. Серед них Є три взаємозамінні протоколи консенсусу у Hyperledger Fabric: Solo, Raft та Kafka.

Solo складається з одного вузла замовника, і його слід використовувати лише для розробки та тестування, оскільки він не забезпечує відмову від аварій.

Raft на основі протоколу Raft реалізує модель "лідер і послідовник", де вузли-лідери обираються на канал. Raft є стійким до аварійних відмов, а це означає, що він може витримати втрату лідера. Рекомендується використовувати у робочих системах, оскільки їх легше налаштувати та керувати порівняно з Kafka.

Kafka забезпечує повноцінний crash-tolerance. Hyperledger Fabric адаптувала Apache Kafka для організації роботи вузлів з метою упорядкування транзакцій. Подібно до Raft, Kafka працює за моделлю «лідер і послідовник», де лідер відповідає за поширення повідомлень підписникам, і якщо вузол-лідер втрачається, то відбувається переобрання нового лідера.

У Кафці замовлення виконує лише лідер, а за лідера можуть проголосувати лише вузли, які синхронізуються з мережею. Це забезпечує стійкість до аварійних відмов, а консенсус досягається за лічені секунди. Хоча Кафка є стійким до аварій

та відмов, він не є BFT, що заважає системі досягти згоди у випадку шкідливих чи несправних вузлів.

Окрім безлічі перевірок підтвердження, валідності та версій, що відбуваються, також проводяться постійні перевірки цифрових особистостей, що відбуваються у всіх напрямках потоку транзакцій. Списки контролю доступу реалізуються на ієрархальних рівнях мережі (замовлення послуги до каналів), а дані транзакцій неодноразово підписуються, перевіряються та засвідчуються на питання автентичності, коли пропозиція транзакцій проходить через різні архітектурні компоненти. Варто розуміти що консенсус не обмежується лише узгодженням та впорядкуванням операцій. Консенсус - це всеохоплююча характеристика мережі, яка досягається як побічний продукт поточних перевірок, що відбуваються під час руху транзакції від пропозиції до реєстру.

Глобальний стан

Глобальний стан – це база даних, що містить кеш поточних значень набору реєстрів. Глобальний стан дозволяє будь-якому додатку швидко отримати доступ до актуальних даних мережі, не вимагаючи обчислення останнього значення шляхом проходження через історію змін стану мережі(журналу транзакцій). Ці дані здебільшого є парами ключ-значення. Глобальний стан часто змінюється, оскільки він складається з сукупності станів реєстрів, що змінюються весь час протягом життя мережі.

По-друге, є блокчейн – реєстр транзакцій, який записує всі зміни, які призвели до поточного світового стану. У блокчейну складна структура: він складається із блоків, що в свою чергу містять транзакції, в яких і відображено зміни в стані блокчейну. Блокчейн містить в собі історичні дані про свій стан в кожен момент починаючи від початкового блоку, і ці історичні дані не можуть бути змінені.

В мережі може бути один або більше окремих блокчейнів, які разом формують глобальний стан мережі.

Глобальний стан містить поточне значення всіх даних як поточного стану всіх реєстрів. Це корисно, оскільки програми зазвичай вимагають поточного значення об'єкта; було б незручно пройти весь блокчейн, щоб обчислити поточне значення об’єкта. Зразок такого стану можна побачити на Рисунку 8.

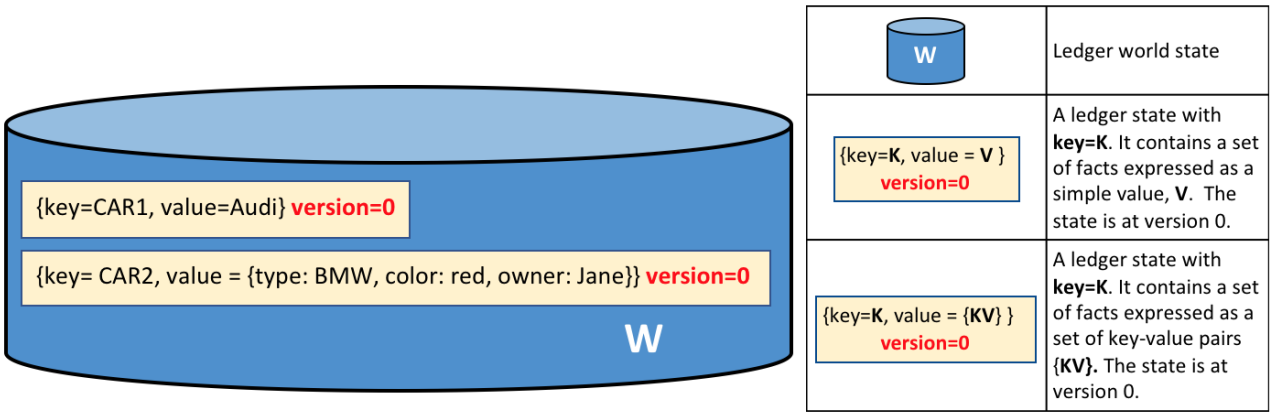

Рисунок 8. Зразок глобального стану блокчейну

Глобальний стан, що складається з двох окремих станів. Перший стан: ключ = CAR1 і значення = Audi. Другий стан має більш складне значення: ключ = CAR2 та значення = {модель: BMW, колір = червоний, власник = Джейн}. Обидва стани знаходяться у версії 0.

Реєстр записує сукупність фактів про певний об'єкт(наприклад, машину). Наведений приклад з CAR1 I CAR2 показує стани для двох автомобілів, CAR1 і CAR2 відповідно. Прикладна програма може посилатися на чейнкод, який може зчитувати дані з реєстрів і отримувати та змінювати значення станів.

Зміна станів в блокчейні здійснюється шляхом створення та подання транзакцій в мережу. Програми формують та надсилають транзакції, сприймаються мережею і після обробки відповідно вносять зміни в окремі атрибути глобального стану, і що в кінцевому рахунку відображається в окремому блокчейні. Важливим є те, що лише транзакції, підписані необхідним набором учасників мережі(організацій), будуть прийняті до внесення змін в глобальний стан. Якщо транзакція не буде затверджена достатньою кількістю учасників, вона

все-рівно буде збережена в блокчейні, але не призведе до зміни глобального стану.

Після створення першого блокчейну в мережі(одразу після першого запуску мережі), глобальний стан є порожнім. Далі будь-яка транзакція, яка має результатом зміну будь-яких даних в мережі, записується на блокчейн. Це дозволяє відтворити глобальному стан мережі з історичних даних блокчейнів на певну одиницю часу.

5.1.2. Структура мережі

Розглянемо найпростіший варіант мережі на базі Hyperledger Fabric. Така мережа N включає одну організацію R4, конфігурацію NC4, ордерер O4(налаштований відповідно до мережевої конфігурації NC4, яка надає адміністративні права організації R4) та Центр Сертифікації CA4(використовується для видачі цифрових сертифікатів адміністраторам та мережевим вузлам організації R4). Візуально найпростіша конфігурації такої мережі повинна була б виглядати як зображено на Рисунку 9.

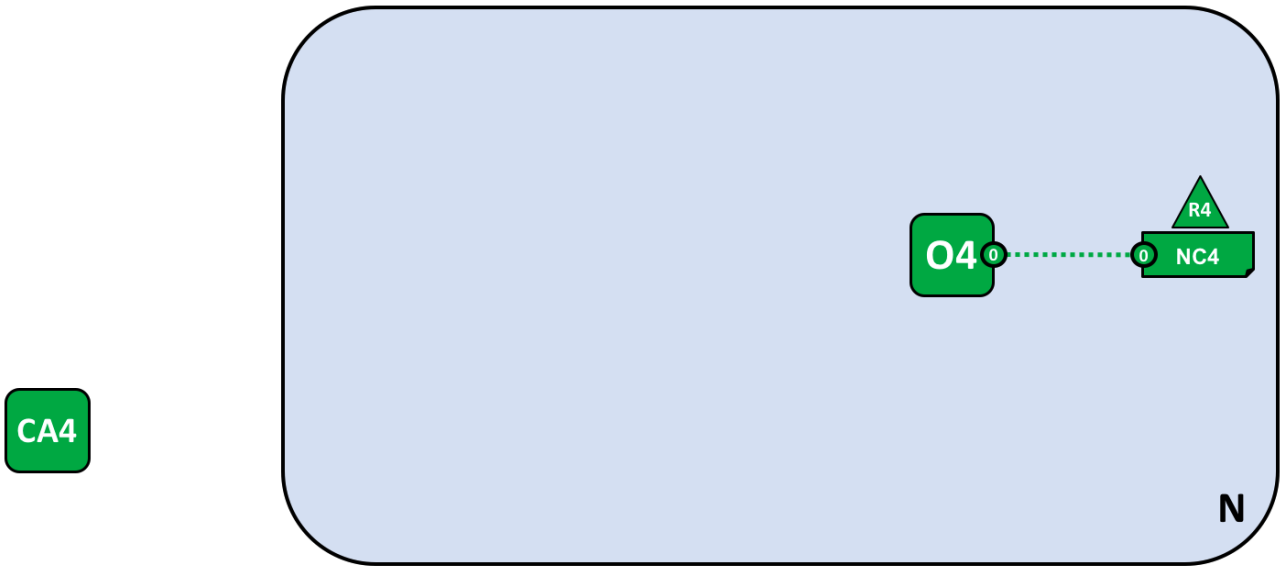

Рисунок 9. Напростіша мережа Hyperledger Fabric, містить лише 1 організацію, конфігурацію мережі, MSP, ордерер та центр сертифікації.

Ордерер

Формування мережі починається із служби впорядкування(ордерер). Корисно думати про ордерер як про початкову точку адміністрування мережі. O4 спочатку налаштовується та запускається адміністратором в організації R4 і розміщується в R4. Конфігурація NC4 містить політику, яка описує стартовий набір адміністративних можливостей мережі. На цьому етапі конфігурація встановлює лише права R4 по мережі.

Центри сертифікації

У мережах даного типу центри сертифікації (напр. CA4) використовується для видачі сертифікатів адміністраторам та мережевим вузлам. CA4 відіграє ключову роль у даній мережі, оскільки він видає сертифікати X.509, які можна використовувати для ідентифікації користувачів, що входять до організації R4. Сертифікати, видані ЦС, можуть також використовуватися для підписання транзакцій, щоб вказати, що організація схвалює результат транзакції - необхідна умова її прийняття до реєстру. Детальніше про ці два аспекти ЦС:

По-перше, різні компоненти мережі blockchain використовують сертифікати, щоб ідентифікувати один одного як представника певної організації. Ось чому зазвичай існує більше одного ЦС, що підтримує мережу блокчейн - різні організації часто використовують різні ЦС. На даному етапі ми використовуємо чотири ЦА в нашій мережі; по одному для кожної організації.

Відображення сертифікатів для організацій-членів здійснюється за допомогою елементу мережі, який називається Провайдером послуг з членства(MSP). Конфігурація мережі NC4 використовує вказаний MSP для ідентифікації сертифікатів, виданих CA4, що дозволяє зв’язувати власників сертифікатів з організацією R4. Потім NC4 може використовувати це ім'я MSP у політиках, щоб надати учасникам конкретних прав R4 над мережевими ресурсами. Прикладом такої політики є визначення адміністраторів R4, які можуть додавати до мережі нових організацій-членів.

По-друге, сертифікати, видані ЦП, лежать в основі процесу генерації транзакцій та перевірок. Зокрема, сертифікати X.509 використовуються при формуванні пропозицій на транзакції та при формуванні відповідей на результати транзакції – всі ці дані підписуються цифровими підписами вузлів мережі, які брали участь в погодженні транзакції. Згодом вузли мережі, які розміщують копії реєстру, підтверджують, що підписи транзакцій є дійсними і належать відповідним вузлам, перед тим як приймати транзакції в реєстр.

Додання мережевих адміністраторів

NC4 спочатку був налаштований так, щоб надати адміністративні права по мережі лише користувачам організації R4. Далі ми дозволяємо користувачам організації R1 також керувати мережею. Візуально це можна побачити на Рисунку 10.

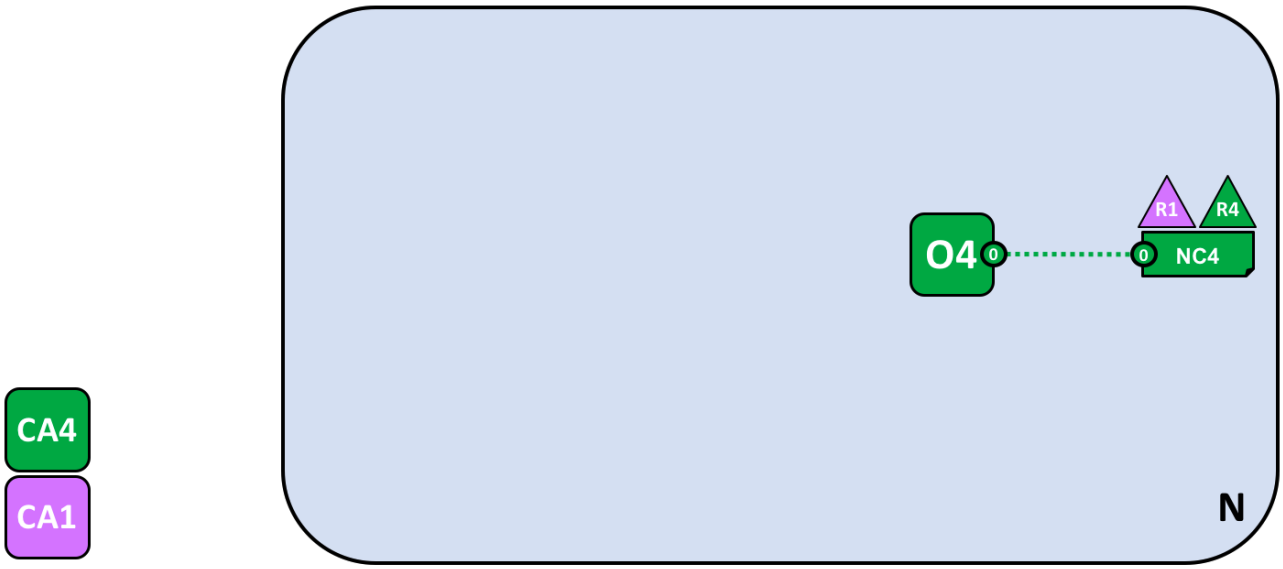

Рисунок 10. До найпростішої мережі додано ще одну організацію R1 та центр сертифікації CA1.

Організація R4 оновлює мережеву конфігурацію так, щоб організація R1 теж була адміністратором. Після цього R1 і R4 мають рівні права щодо конфігурації мережі. Далі додається центр сертифікації CA1 - він може використовуватися для ідентифікації користувачів з організації R1. Після цього користувачі як R1, так і R4 можуть управляти мережею. Для демонстрації

адміністративних повноважень в мережу також додається організація R2, якій не надаються ніякі додаткові права.

Хоча вузол замовлення, O4, працює на інфраструктурі R4, R1 має спільні адміністративні права на нього, якщо він може отримати доступ до мережі. Це означає, що R1 або R4 могли оновити конфігурацію мережі NC4, щоб дозволити організації R2 підмножину мережевих операцій. Таким чином, навіть незважаючи на те, що R4 працює з сервісом замовлення, і R1 має повні адміністративні права на неї, R2 має обмежені права на створення нових консорціумів(через відсутність адміністративних прав).

У своїй найпростішій формі ордерер - це єдиний вузол у мережі. Ордерери такоє можуть бути багатовузлові і налаштовані на різні вузли в різних організаціях. Наприклад, ми можемо запустити O4 в R4 і з'єднати його з O2, окремим вузлом замовлення в організації R1. Таким чином, ми мали б багатосторонню структуру адміністрування.

Консорціум

Незважаючи на те що R1 І R4 мають адміністративні права над мережею, для розкриття її повного функціоналу необхідно створити консорціум.

Консорціум є логічним об’єднанням організацій в мережі для укладання між ними мережевих взаємозв’язків. З бізнесової точки зору в консорціуми об’єднують організації які мають спільну мету та мають намір комуні кувати в межах мережі. З технічної точки зору консорціум організацій необхідний для створення важливої частини блокчейна Hyperledger Fabric - каналу.

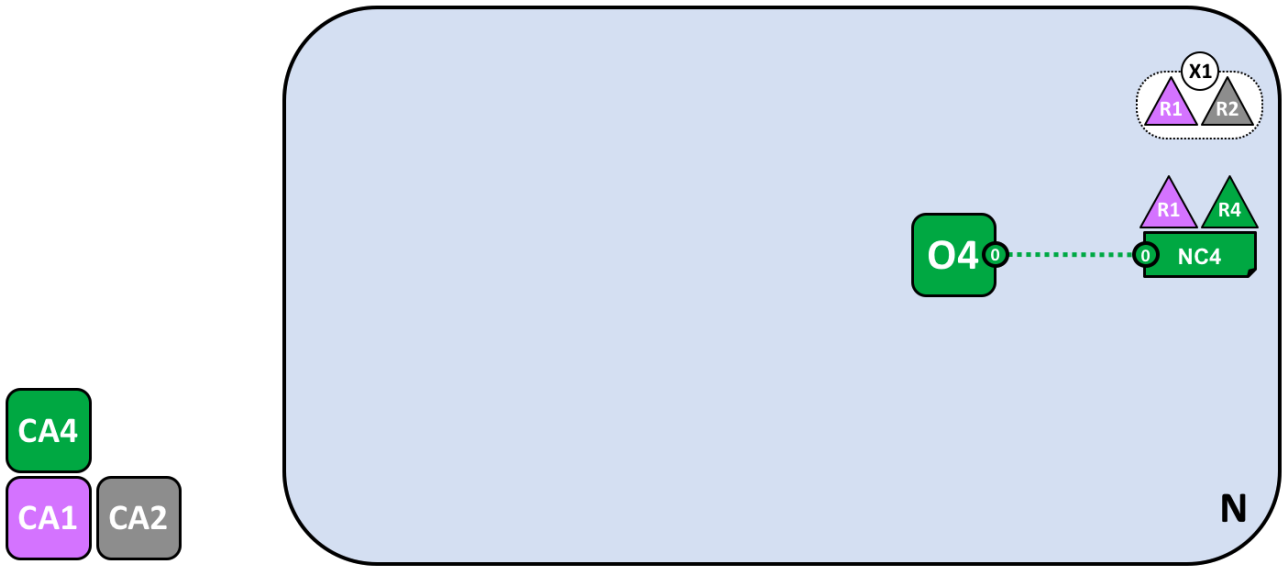

Рисунок 11. Додавання консорціуму X1 та об’єднання в ньому організацій R1 та R2

Адміністратор мережі визначає консорціум X1, який містить два члени, організації R1 і R2. Це визначення консорціуму зберігається в мережевій конфігурації NC4 і буде використовуватися на наступному етапі розробки мережі. CA1 та CA2 є відповідними органами сертифікації для цих організацій.

Через налаштування NC4, лише R1 або R4 можуть створювати нові консорціуми. Ця діаграма показує додавання нового консорціуму X1, який визначає R1 і R2 як його складові організації. Також ми можемо побачити, що CA2 додано для ідентифікації користувачів з R2. Зауважте, що консорціум може мати будь-яку кількість членів організації - ми тільки що показали двох, оскільки це найпростіша конфігурація.

Створення каналу для консорціуму

Канал - це первинний механізм комунікацій, за допомогою якого члени консорціуму можуть спілкуватися один з одним. У мережі може бути один або кілька каналів.

Канали забезпечують механізм приватного спілкування та приватних даних між членами консорціуму. Канали забезпечують конфіденційність з інших каналів та з мережі. Тому, якщо різні консорціуми всередині мережі потребуватимуть

відповідного обміну інформацією та процесами, канали - забезпечують ефективний механізм для цього. Тобто канали забезпечують ефективний обмін інфраструктурою, зберігаючи конфіденційність даних та комунікацій.

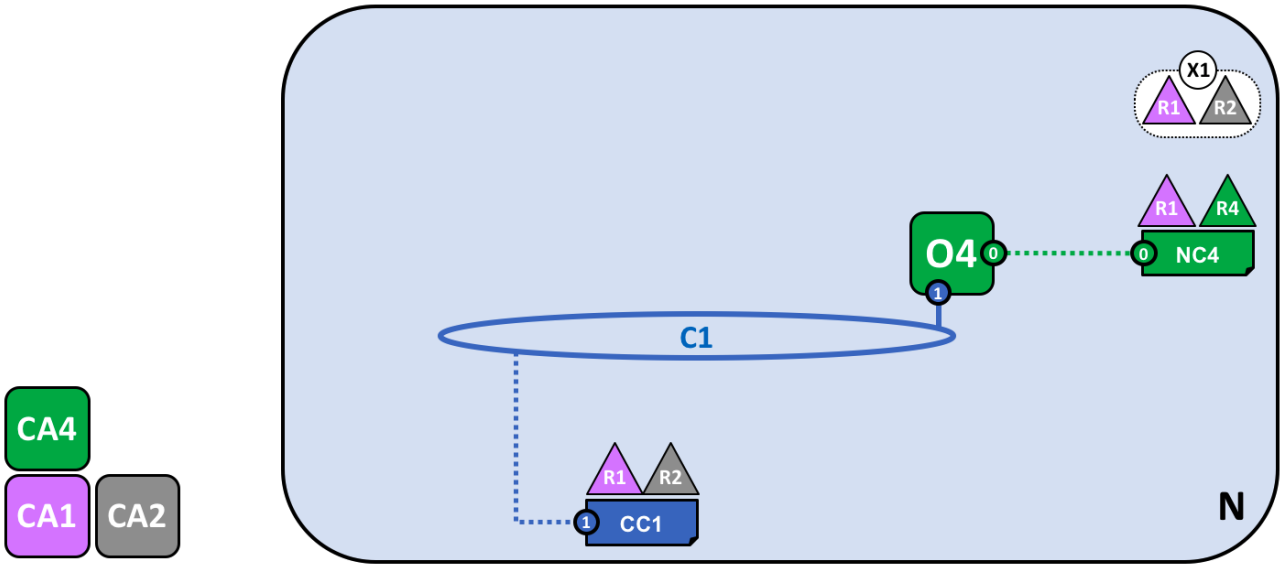

Рисунок 12. Додання каналу C1

Для R1 та R2 був створений канал C1 з використанням консорціуму X1. Канал керується конфігурацією каналу CC1, що є повністю окрема від мережевої конфігурації NC4. CC1 управляється R1 і R2, які мають рівні права над C1. R4 взагалі не має прав на CC1.

Технічно в канал може бути підключена будь-яка кількість організацій, але наразі для простоти та демонстрації до каналу підключені лише 2 організації. Даний канал C1 дозволяє комунікацію лише організацій R1 і R2, а організації R3 та R4 не мають до нього доступу. Конфігурація CC1(конфігурація каналу C1) містить політику, яка регулює права R1 та R2 над каналом C1, а R3 та R4 не мають дозволів на цьому каналі. R3 і R4 можуть взаємодіяти з C1, лише якщо вони будуть додані кимось з існуючих учасників каналу до відповідної політики конфігурації.

Канал C1 забезпечує приватний механізм зв'язку для консорціуму X1. Канал C1 підключений до ордерера O4, але нічого іншого до нього не приєднано. На базі цього каналу надалі в мережу можуть додаватися клієнтські додатки та вузли.

Навіть незважаючи на те, що канал C1 є частиною мережі N, він є її відокремленим елементом. Контроль над ним мають лише організації, які прямо

вказані в конфігурації каналу. Аналогічно, будь-які оновлення мережевої конфігурації NC4 не матимуть прямого впливу на конфігурацію каналу CC1; наприклад, якщо визначення консорціуму X1 буде змінено, це не вплине на членів каналу C1. Тому канали корисні, оскільки дозволяють приватні комунікації між організаціями, що складають канал. Більше того, дані каналу повністю ізольовані від решти мережі, включаючи інші канали.

Крім того, існує також спеціальний системний канал, визначений для використання службою замовлення. Він поводиться точно так само, як звичайний канал, який іноді з цієї причини називають каналами додатків.

5.1.3. Мережеві вузли

Після додавання в мережу вузла P1 в мережі нарешті з'являється перша копія блокчейну. Копія блокчейн-реєстру L1 фізично розміщена на вузлі P1, а логічно - на каналі C1. Через канал C1 вузол P1 спілкується з ордерером O4.

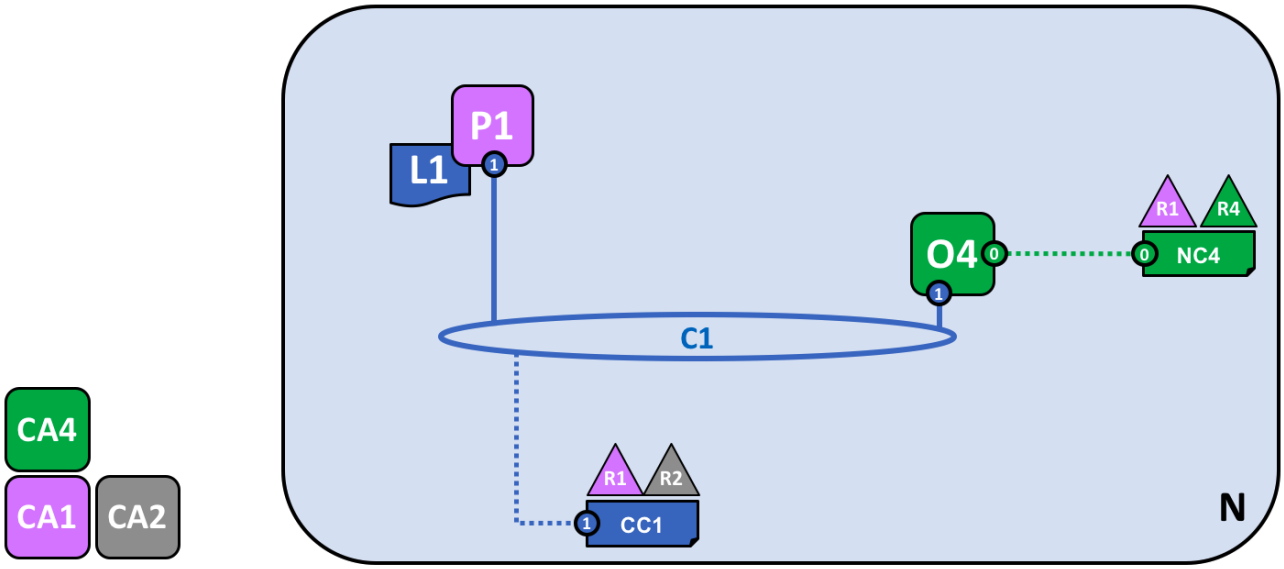

Рисунок 13. До каналу C1 приєднано перший вузол P1 з реєстром L1.

Вузли мережі(peer) - це мережеві компоненти, де розміщуються копії реєстру блокчейн. Призначення P1 в мережі полягає лише в тому, щоб розмістити копію реєстру L1 для доступу інших компонентів мережі.

Ключовою частиною конфігурації P1 є ідентифікатор X.509, виданий CA1, який асоціює P1 з організацією R1. Після запуску P1 він може приєднатися до

каналу C1 за допомогою ордерера O4. Коли O4 отримує цей запит на приєднання, він використовує конфігурацію каналу CC1 для визначення дозволів P1 на цьому каналі. В нашому випадку CC1 визначає що P1 може читати та записувати інформацію в реєстрі L1.

Вузли приєднуються до каналів організаціями, яким вони належать. В каналі може бути кілька вузлів що належать одній або різним організаціям. Також у різних вузлів можуть бути різні ролі в мережі.

5.1.4. Смарт-контракти

Основна користь усієї вище переліченої інфраструктури мережі розкривається при підключенні програм та смарт-контрактів, що взаємодіють з мережею.

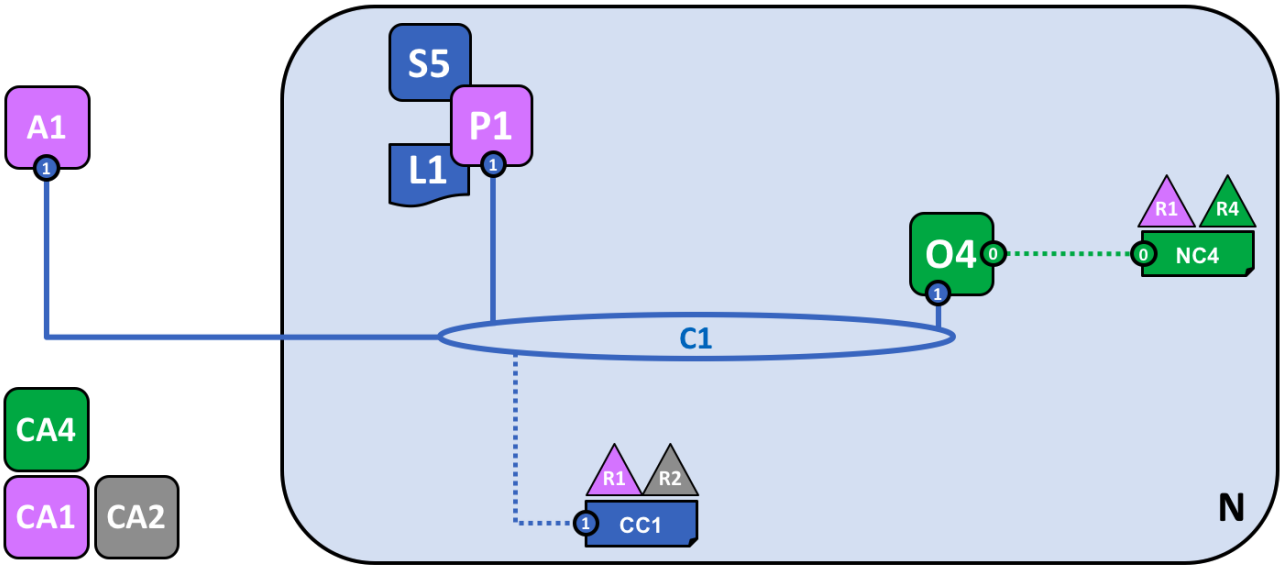

Рисунок 14. До мережі додано додаток A1, що підключений до каналу C1, та смарт-контракт S5, що встановлено на вузлі P1.

На P1 встановлено смарт-контракт S5. Клієнтська програма A1 в організації R1 може використовувати S5 для доступу до реєстру через мережевий вузол P1. A1, P1 і O4 приєднані до каналу C1, тобто всі вони можуть використовувати засоби зв'язку, що надаються цим каналом.

Клієнтська програма A1 може використовувати канал C1 для підключення до конкретних мережевих ресурсів - в цьому випадку A1 може підключитися як до вузла P1, так і до вузла ордерера O4. Як і всі вузли та ордерери, клієнтська

програма матиме ідентифікацію, яка асоціює її з організацією. У нашому випадку клієнтська програма A1 асоціюється з організацією R1.

Клієнтська програма розміщена та працює окремо від мережі, але вузол (в даному випадку це P1) дозволяє їй підключатись та взаємодіяти з мережею(а точніше з усім, що підключене до каналу C1).

По-факту додаток A1 підключається до вузла P1, і може взаємодіяти з мережею через смарт-контракт S5, що встановлений на вузлі P1. Смарт-контракт S5 надає чітко визначений набір способів, за допомогою яких A1 може звертатися до реєстру L1, наприклад які саме частини поточного стану реєстру можна зчитувати та модифікувати.

Смарт-контракти створюються розробниками додатків у кожній організації для впровадження бізнес-логіки, спільної для членів консорціуму. Смарт-контракти використовуються для створення транзакцій, які згодом можуть бути розповсюджені на кожен вузол мережі. Для початку роботи смарт-контракту в мережі, його спершу потрібно встановити на певні один або кілька вузлів мережі(до яких будуть звертатися клієнтські програми, що будуть викликати функції смарт-контракту), а далі ініціювати його в каналі.

*Встановлення смарт-контракту. С*март-контракт S5 встановлюється на вузол P1 адміністратором організації R1, після чого P1 має повні відомості про S5. Зокрема, P1 може бачити логіку реалізації S5 - програмного коду, який він використовує для доступу до реєстру L1. Якщо в організації є декілька вузлів у каналі – не потрібно встановлювати смарт-контракт на кожен вузол, достатньо буде встановити лише на одному або кількох.

Ініціація смарт-контракту. Далі, після встановлення S5 на одному або кількох вузлах, перш ніж всі учасники каналу взнають про існування смарт-контракту та зможуть з ним взаємодіяти, його потрібно ініціювати в каналі. У нашій мережі, в якій смарт-контракт встановлений лише на вузлі P1, адміністратор в організації R1 інстанціює S5 на каналі C1 за допомогою P1. Після ініціації кожен учасник каналу C1 зможе взаємодіяти з смарт-контрактом S5, отже тепер його методи можна викликати за допомогою додатку A1.

Надзвичайно важливим елементом ініціації смарт-контракту в каналі є політика схвалення операцій смарт-контрату. Вона описує, які організації повинні затверджувати транзакції, перш ніж ці транзакції будуть додаватися до блокчейну і відповідно зберігатися в копіях реєстру всіх вузлів каналу. У нашій мережі транзакції можуть бути прийняті в регістр L1 лише тоді, коли R1 або R2 їх затверджують.

Акт ініціації встановлює політику схвалення в конфігурацію каналу CC1; це дозволяє йому отримати доступ до будь-якого учасника каналу.

Викликання розумного контракту.

Після встановлення розумного контракту на рівний вузол та інстанціювання на каналі, клієнтська програма A1 може його може викликати. Клієнтські додатки роблять це, надсилаючи пропозиції щодо транзакцій вузлам, що належать організаціям, визначеним політикою затвердження смарт-контракту. Пропозиція про транзакцію служить вхідним кодом до смарт-контракту, який використовує його для створення схваленої відповіді на транзакцію, яка повертається вузлом клієнтській програмі.

Саме ці відповіді на транзакції упаковуються разом із пропозицією про транзакцію, щоб сформувати цілком схвалену транзакцію, яку можна поширити по всій мережі.

На цьому етапі розвитку мережі ми бачимо, що організація R1 повністю бере участь у мережі. Її програми - починаючи з A1 - можуть отримати доступ до книги L1 за допомогою смарт-контракту S5, щоб генерувати транзакції, які будуть затверджені R1, і тому приймаються в реєстр, оскільки вони відповідають політиці затвердження.

Варто звернути увагу на те, що хоча кожен компонент каналу тепер може отримати доступ до S5, вони не в змозі бачити його програмну логіку. Це залишається приватним для тих вузлів, на які його було встановлено; в нашому випадку це стосується P1. Концептуально це означає, що це смарт-контракт представляє собою програму лише для P1, а для решти учасників каналу він є просто інтерфейсом без логіки.

Як бачимо існує два різних види рівних вузлів; ті, на яких встановлено смарт-контракт, і ті, на якиз не встановлено. Програма може викликати методи смарт-контракту лише якщо він встановлений на вузлі, до якого підключено програму, але вона може знати про інтерфейс смарт-контракту, підключившись до каналу.

Незалежно від факту встановлення на вузлі смарт-контракту, кожен вузол бере участь якщо не в виконанні самого коду контракту(як це роблять вузли де цей контракт встановлений), то принаймні беруть участь в формуванні транзакції яка далі приймається блокчейном. Це зумовлено тим що всі вузли можуть перевірити та згодом прийняти або відхилити транзакції до їх копії реєстру L1. Однак лише вузли із встановленим смарт-контрактом можуть брати участь у процесі затвердження транзакцій, що є центральним у створенні дійсних транзакцій.

Види вузлів

У Hyperledger Fabric вузли можуть брати на себе декілька ролей, залежно від налаштування мережі.

- Вузол зберігання(Committing peer). Кожен вузол в каналі є цим вузлом зберігання. Він отримує блоки згенерованих транзакцій, які згодом перевіряються перед тим як бути доданими до локальної копії блокчейну.
- Вузол схвалення. Кожен вузол із встановленим смарт-контрактом може потенційно стати вузлом схвалення. Однак, щоб даний вузол справді став вузлом схвалення, клієнтський додаток повинен використовувати саме цей вузол для виклику методу смарт-контракту і для створення відповіді на транзакцію, що підписано цифровим підписом цього вузла, що означає що цей вузол схвалив дану транзакцію. Політика схвалення смарт-контракту визначає організації, чий вузол повинен підписати згенеровану транзакцію, перш ніж її можна буде прийняти на копію реєстру цього вузла.
- Вузол-лідер. Коли в організації є декілька вузлів у каналі, вузол-лідер - це вузол, який бере на себе відповідальність за розповсюдження транзакцій від

ордерера до інших вузлів організації. Вузол може бути обраним як лідер в процесі статичного чи динамічного вибору лідера. В першому випадку певний вузол завжди обирається як лідер, в другому – лідер постійно змінюється. Це означає також що вузли організації можуть мати одного або декількох лідерів, підключених до оредера. Це може допомогти підвищити стійкість та масштабованість у великих мережах, які обробляють великі обсяги транзакцій.

- Якірний вузол. Такий вид вузла використовується тоді, коли вузлу однієї організації потрібно зв’язатись з вузлом іншої. У такому випадку звертаються в першу чергу до вузла, який зазначений як якорний в конфігурації організації. Організація може мати нуль або кілька якорних вузлів.

Варто зауважити що один вузол може бути одночасно вузлом зберігання, вузлом схвалення, вузлом-лідером і якірним вузлом.

5.2. Смарт-контракт “DTL”

Смарт-контракт в мережі Hyperledger Fabric – це програма, що імплементує бізнес-логіку в програмний код, написана, встановлена та інстранційована в каналі мережі, до якої можуть звертатися користувачі за викликом функцій. Смерт-контракт визначає правила взаємодії між різними організаціями у виконуваному коді.

Словосполучення «смарт-контракт» було створено комп'ютерним вченим Ніком Сабо в 1996 році, для підкреслення того, що він називає «високорозвинені практики» договірного права і пов'язаних з діловою практикою в розробці електронних протоколів торгівлі, між незнайомими людьми в Інтернеті. У 1996 році Сабо описував його так:

«Нові інституції і нові способи формалізації відносин цих інституцій сталі можливі завдяки цифровій революції. Я називаю ці контракти «розумними» тому що вони набагато більш функціональні, ніж їхні неживі паперові предки. Не передбачається використання штучного інтелекту. Смарт-контракти це набір обіцянок у цифровому форматі, включно з протоколами за якими сторони виконують ці обіцянки.»

Смарт-контракти використовуються здебільшого для розуміння того, що відбувається на блокчейні. У цій інтерпретації смарт-контракт не обов'язково має відношення до класичної концепції договору, але може бути будь-якою комп'ютерною програмою.

В межах Hyperledger Fabric терміни смарт-контракту і чейнкоду часто ввикористовують взаємозамінно. Загалом, смарт-контракт визначає логіку транзакцій, яка контролює життєвий цикл бізнес-об’єкта, що міститься у глобальному стані. Він упаковується у чейнкод, який потім розгортається в блокчейн-мережі. Слід вважати смарт-контракти засобом управління транзакціями, тоді як чейнкод визначає, як смарт-контракти пакуються для розгортання в мережі.

Смарт-контракти в першу чергу змінюють, отримують та видаляють стани у глобальному стані, а також можуть запитувати незмінний блокчейн-запис про транзакції. Критично, що в усіх випадках, незалежно від того, чи створюються транзакції читання, оновлення чи видалення бізнес-об'єктів у глобальному стані, блокчейн містить незмінний запис цієї операції.

Надзвичайно важливою частиною роботи смарт-контракту є політика схвалення: вона вказує, які організації в блокчейн-мережі повинні схвалену транзакцію, згенеровану даним смарт-контрактом, щоб ця транзакція була визнана дійсною.

Наприклад, в політиці схвалення можна визначити, що три з чотирьох організацій, які беруть участь у блокчейн-мережі, повинні були схвалити транзакцію до того, як вона буде визнана дійсною. Усі транзакції, дійсні чи недійсні, додаються до розподіленої книги, але лише дійсні транзакції оновлюють світовий стан.

Концепція політики схвалення - це те, що відрізняє Hyperledger Fabric від інших реалізацій технології блокчейн, таких як Ethereum або Bitcoin. У цих системах дійсні транзакції можуть генеруватися будь-яким вузлом у мережі. Hyperledger Fabric розрахований на використання в корпоративних мережах де часто зберігаються конфіденційні дані, тому транзакції повинні бути підтверджені

надійними організаціями в мережі. Наприклад, урядова організація повинна підписати дійсну операцію з видачею ідентифікації, або ж покупець і продавець автомобіля повинні підписати угоду з передачі автомобіля. Політика схвалення розроблена таким чином, що дозволяє Hyperledger Fabric краще моделювати такі типи взаємодій у реальному світі.

DTL

В програмній реалізації даного продукту було розроблено смарт-контракт, що включає логіку децентралізованого реєстру правочинів(що відображено в його назві, “**D**ecentralized **T**reaty **L**edger”). Таким чином, логіка смарт-контракту дозволяє створювати, змінювати та зчитувати дані про комерційні договори в мережі блокчейн.

Смарт-контракт може використовуватись в наступному сценарії: припустимо що два користувача мережі U1 та U2, що відповідно відносяться до організацій O1 та O2, досягли угоди щодо купівлі-продажу земельної ділянки. Для перетворення їх домовленості у форму дійсного договору їм необхідно написати умови угоди в електронній формі та завантажити у реєстр, де текст угоди буде надійно зберігатись. Для цього вони можуть використовувати смарт-контракт DTL, який дозволяє створювати в блокчейні записи про договори з наступними атрибутами:

- заголовок(Title) – строка, яка містить назву договору;
- опис(Description) – строка, що містить короткий опис договору(суто для полегшення каталогування);
- текст(Text) – сам текст договору, як правило форматований;
- сторона 1(Signatory 1) – ПІБ або інші ідентифікатори сторони договору;
- сторона 2(Signatory 2) – ПІБ або інші ідентифікатори сторони договору;
- час створення(timestamp) – час збереження договору в мережі;

- підписано стороною 1(signed1) – булеве значення, позначає чи підписала сторона 1 цей договір;
- підписано стороною 2(signed2) – булеве значення, позначає чи підписала сторона 2 цей договір.

ДОГОВІР КУПІВЛІ-ПРОДАЖУ
земельної ділянки

місто Нова Одеса, Миколаївської області, _____**дві тисячі чотирнадцятого року.**

Ми, ті, що підписалися нижче, Прохоров Володимир Сергійович, який проживає в м. Миколаєві, пр. Жовтневий, 6, кв. 60, реєстраційний номер облікової картки платника податків за даними Державного реєстру фізичних осіб – платників податків 2325301936, свідоцтво № 1450, видане Міністерством юстиції України 08 липня 2013 року який діє на підставі Постанови Господарського суду Миколаївської області від 17 січня 2013 року, справа № 5016/2954/2012(2/30), як ліквідатор банкрута - фізичної особи-підприємця, Барни Федора Петровича, що мешкає за адресою: Миколаївська область, Новоодеський район, с. Гуріївка, вул. Набережна, 62, реєстраційний номер облікової картки платника податків за даними Державного реєстру фізичних осіб – платників податків 2725615071, з однієї сторони, надалі за текстом – **Продавець**, та_____, з другої сторони, надалі за текстом – **Покупець**, діючі вільно, цілеспрямовано, свідомо і добровільно, розумно та на власний розсуд, без будь-якого примусу, як фізичного, так і психічного, бажаючи реального настання правових наслідків обумовлених нижче, перебуваючи при здоровому розумі та ясній пам'яті, усвідомлюючи значення своїх дій та керуючи ними, попередньо ознайомлені з приписами цивільного законодавства, що регулюють укладений договір, керуючись главою 54 Цивільного кодексу України, вільно володіючи українською мовою та домовившись про укладення цього Договору за місцем знаходження майна, уклали цей Договір про таке:

1. Предмет договору

1.1. Продавець зобов'язується передати у власність Покупцю майно – **земельну ділянку загальною площею 0,1500 га, за адресою Миколаївська область, Новоодеський район, с. Гур'ївка, вулиця Набережна, 91 (дев'яносто один**), а Покупець зобов'язується прийняти майно та сплатити ціну відповідно до умов, визначених вищевказаним протоколом проведення аукціону. **Кадастровий номер земельної ділянки – 4824882000:06:058:0001**.

Цільове призначення земельної ділянки: для індивідуального житлового, гаражного і дачного будівництва. Категорія земель: землі житлової та громадської забудови. Вид використання земельної ділянки: для будівництва і обслуговування житлового будинку, господарських будівель і споруд (присадибна ділянка). Форма власності: приватна власність. Земельна ділянка, згідно з Державним актом на право власності на земельну ділянку, має такий опис меж : від А до Б землі державної власності; від Б до В землі Барни Ф.П.; від В до А землі загального користування.

1.2. Вищевказана земельна ділянка належить Барні Федору Петровичу на підставі Державного акта на право власності на земельну ділянку серії МК № 002476, виданого Гур'ївською сільською радою 12 лютого 1998 року, зареєстрованого в Книзі записів державних актів на право приватної власності на землю за № 1234. Право власності зареєстроване в Державному реєстрі речових прав на нерухоме майно _____ року, номер запису про право власності _____, реєстраційний номер об'єкта _____.

2. Ціна

2.1. Продаж майна на підставі Протоколу аукціону з продажу майна банкрута № ___, який складено Організатором аукціону - Універсальною біржею «Південь» 14.01.2014 року вчиняється за_____гривень

Рисунок 15. Зразок договору купівлі-продажу земельної ділянки[16].

Таким чином, договір може бути збережений в мережі в наступному вигляді:

Заголовок: «ДОГОВІР КУПІВЛІ-ПРОДАЖУ»

Опис: «Договір між сторонами ОСОБА1 та ОСОБА2 про відчуження об'єкту нерухомості»

Текст: «Ми, ті, що підписалися нижче, Прохоров Володимир Сергійович, який проживає в м. Миколаєві, пр. Жовтневий, 6, кв. 60, реєстраційний номер облікової картки…»

Сторона1: «Прохоров Володимир Сергійович»

Сторона2: «Барна Федір Петрович»

Час створення: 1571947052 Підписано стороною 1: true Підписано стороною 2: true

Таким чином, смарт-контракт вимагає спеціального формату подання даних в його методи.

Смарт-контракт має наступні методи:

- createContract
- getContract
- getAllContracts
- signContract

Детальніше про методи смарт-контракту далі.

CreateContract

Даний метод використовується для запису нових договорів в децентралізованому реєстрі. Він вимагає наступні параметри:

- contractId (строка для ідентифікації договору смарт-контрактом, наприклад «CONTRACT01»)
- title
- description
- signatory1
- signatory2
- timestamp

При виклику цього методу параметри, що були передані, обробляються та зберігаються в стан блокчейну. Технічно це реалізовується наступним чином:

При створенні смарт-контракту в блокчейні, в глобальному стані(який, як було описано раніше, є базою даних) виділяється об’єм пам’яті(визначається відповідним простором імен в базі даних), що далі використовується саме смарт-контрактом для збереження його відповідних даних. Для зручності про це сховище варто думати як про асоціативний масив. При виклику методу

createContract смарт-контракту, в цьому сховищі створюється нова пара ключ-значення, де ключем є строка, передана в виклик методу як параметр contractId, а значенням є новий внутрішній асоціативний масив, який в свою чергу містить пари ключ-значення усіх інших параметрів виклику функції.

Тому після виклику цього методу, формування транзакції та її підтвердження, глобальний стан мережі в частині смарт-контракту DTL буде виглядати приблизно так:

```
{
    "DTL":{
            "CONTRACT01":{
                    "title":"ДОГОВІР КУПІВЛІ-ПРОДАЖУ",
                    "description":"Договір між сторонами ОСОБА1 та ОСОБА2 про відчуження об’єкту нерухомості",
                    "text":"Ми, ті, що підписалися нижче, Прохоров Володимир Сергійович, який проживає в м. Миколаєві, пр. Жовтневий, 6, кв. 60, реєстраційний номер облікової картки…",
                    "signatory1":"Прохоров Володимир Сергійович",
                    "signatory2":"Барна Федір Петрович",
                    "timestamp":"1571947052",
                    "signed1":false
                    "signed2":false
            }
    }
}
```

Варто звернути увагу на те, що при виклику методу смарт-контракту ми не передавали параметри signed1 I signed2, тому вони в збереженому договорі мають мулеве значення за-замовчуванням «false».

GetContract і GetAllContracts

Дані методи використовуються для зчитування даних про збережені договори. Вони відрізняються лише тим що GetContract приймає параметр contractId та відповідно виводить лише дані про договір, який записаний в сховищі під ключем що дорівнює значенню параметра contractId(наприклад, “CONTRACT01”) і виводить асоціативний масив, що зберігається в сховищі під цим ключем. Зразок того, як виглядає вивід цієї функції, було показано при описі методу CreateContract.

GetAllContracts по-суті робить те саме, але не отримує параметри і просто виводить усі збережені в сховищі договори.

SignContract

Цей метод використовується для того, щоб відобразити в даних договору факт того, що він був підписаний конкретною стороною. Він очікує 2 параметри – contractId та signatory. contractId – це ідентифікатор договору, з яким здійснюється операція підписання, а signatory – це строка, що містить ім’я підписанта. Коли даний метод викликається, код спершу перевіряє чи існує в сховищі договір з таким ідентифікатором і якщо так – перевіряє чи одна з сторін договору співпадає зі значенням параметру signatory. Якщо значення співпадають – визначається якою саме стороною(signatory1 чи signatory2) підписується договір, і відповідно присвоюється значення true змінній signed1 або signed2(якщо параметр signatory дорівнює signatory1 – змінюється значення signed1, і навпаки коли signatory дорівнює signatory2 – змінюється signed2).

Після затвердження транзакції з викликом цього методу, відображення договору в сховищі змінюється і його атрибут signed1 або signed2 відповідно відображаються як true.

Якщо ж параметр signatory не дорівнює жодній з сторін договору то код повертає помилку виконання смарт-контракту і зміни в договір не вносяться.

5.3 Серверний додаток API

Після розгортання мережі Hyperledger Fabric та встановлення смарт-контракту DTL, користувачі можуть починати взаємодіяти з децентралізованим реєстром правочинів. В найпростішому варіанті для цього потрібні:

- сервер, на якому буде встановлено вузол мережі
- криптографічні матеріали, що були надані користувачу при реєстрації в мережі
- термінал для підключення до консолі вузла

В такій конфігурації користувач може використовувати функціонал децентралізованого реєстру правочинів і взаємодіяти з смарт-контрактом для збереження, відтворення або підписання договорів зі сховища смарт-контракту. Однак варто зазначити, що для для того щоб просто викликати метод смарт-контракту DTL за допомогою базових інструментів, що постачаються Hyperledger Fabric, потрібно буде відкрити термінал операційної системи сервера, на якому розташовано вузол, і ввести наступну команду:

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n DTL --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["signContract","CONTRACT01",” Прохоров Володимир Сергійович”]}'

В даній команді зазначається джерело отримання криптографічного підпису користувача, адреса вузла для звернення, ім’я смарт-контракту для виклику, його конкретний метод, параметри. Як бачимо команда є дуже довгою і вручну вводити її є дуже непрактичним, особливо враховуючи що функціоналом децентралізованого реєстру правочинів повинні мати змогу користуватись і користувачі, що не володіють такими технічними навиків. Саме тому для спрощення доступу та взаємодії до функціоналу DTL було вирішено створити API, що буде отримувати HTTP запити, перероблювати їх у команди належного формату та змісту та перенаправляти їх на відповідний вузол мережі.

Це API було реалізовано в формі серверного додатку на мові NodeJS з використанням бібліотеки express(для швидкого розгортання серверу) та Javascript бібліотекою “fabric-network”, що розповсюджується як безкоштовний продукт для легшої інтеграції додатків з Hyperledger Fabric.

5.3.1. Основні модулі API

Основними структурними елементами API є:

- environment. У змінних оточення визначається адреса вузла Hyperledger Fabric, токени авторизації вхідних і вихідних http-запитів, місцезнаходження криптографічних матеріалів, порти сервера та інші налаштування. Змінні оточення лежать в папці *config*.
- Director – цей модуль запускається першим при запуску додатку. Він надсилає команди ApplicationBuilder-у формувати окремі структурні елементи додатку і запустити їх.
- ApplicationBuilder - даний модуль є тим, що формує решту програми та підключає до неї усі необхідні модулі, такі як ApiBuilder та всі його складові, логери, валідатори, обробники помилок, тощо. Використовуючи конфігурацію, що зберігається в файлі config/moduleDependencies.js, ApplicationBuilder отримує інформацію про те які модулі повинні бути додані до додатку, а також від яких інших модулів вони залежать для своєї належної роботи.

- ApiBuilder – цей робить даний додаток справжнім HTTP API. Він запускає прослуховування HTTP запитів, що надходять на певний порт, отримує всі запити на цей порт, розбирає запит згідно його URL та параметрам та перенаправляє на відповідні обробники, в яких вже виконується логіка обробки запитів сервісом DTLService.
- HttpProvider – даний модуль вміщує в собі логіку взаємодії з іншими елементами системи через HTTP запити. По-суті містить в собі функціонал для надсилання GET і POST запитів із відповідним корисним навантаженням даних.
- HttpRequestEndpoints - цей модуль містить в собі інформацію про адреси url, на які сервер буде слухати HTTP запити з певним методом(GET,POST,PUT,DELETE) та перенаправляти отриманий запит на відповідні обробники запитів в модулі HttpRequestHandlers.
- HttpRequestHandlers – цей модуль містить в собі набір обробників, що викликаються API при надходженні на певну адресу HTTP запитів з даними. Здебільшого в обробниках викликаються функції модуля DTLService.
- DTLService – даний модуль включає в себе саму логіку перетворення набору параметрів з HTTP запиту у виклик команди смарт-контракту. даний модуль вміщує в собі основу логіки всього додатку. Він виконує велику кількість функцій – зчитує криптографічні матеріали користувача, підключається до мережі Hyperledger Fabric, підключається до відповідного смарт-контракту, формує transaction proposal що далі надсилається в мережу та повертає результати виконання транзакції або відловлює помилки в процесі обробки транзакції. Так, він використовує CredentialManager для отримання криптографічних даних користувача з файлової системи або з іншого сховища секретів(Hashicorp Vault). Окрім цього даний модуль включає методи createContract,updateContract, signContract, deleteContract, що відповідно дозволяють створювати, оновлювати, підписувати та видаляти договори.
- Fabric-sdk - software-development-kit, розповсюджуваний як вільне ПЗ, що включає функціонал з взаємодії з елементами системи Hyperledger Fabric, такими як вузли.

- CredentialManager – даний модуль дозволяє зчитувати критичні дані доступів та приватних ключів з різних сховищ даних(в даній системі для простоти використовується файлова система машини-хоста).

- Authrorizer – даний модуль дозволяє контролювати те щоб усі запити, що приходять на порт серверного додатку, були належно авторизовані, відповідно до записаних в змінних оточення авторизаційних токенів та заголовків http-запитів, що отримуються та надсилаються додатком.

5.3. 2.Алгоритм запуску та роботи додатку

Між запуском додатку командою *npm start* та початком його повноцінної роботи відбувається процедура побудови та компонування додатку. В архітектурі додатку часто використовується паттерн «Будівельник»(англ. Builder), який передбачає існування спеціального об’єкту-будівельника, що покроково будує продукт.

```
build(){
    this.createApplication();
    this.bootstrapApplication();
    this.createModuleList();
    await this.composeApplicationModules();
        return resolve(this.getApplication());
   }
```

Метод this.createApplication() створює пустий об’єкт класу Application та присвоює його в змінну this.application будівельника. Далі запускається метод bootstrapApplication, що додає у додаток такі компоненти як Logger(забезпечує легування додатку), Validator(дозволяє перевіряти змінні на відповідність певним типам даних та атрибутам), endpointHelper(допомагає формувати адреси для прослуховування вхідних HTTP-запитів), urlMap(допомагає формувати шляхи для вихідних HTTP-запитів). Наступним методом createModuleList() будівельник

зчитує дані з файлу config/moduleDependencies.js. Далі наведено вміст цього файлу:

```
module.exports = {
  [GENERAL_MODULES]:{
      'apiBuilder':add(ApiBuilder,['server','dtlService']),
      'httpService':add(HttpService,[
      ]),
      'dtlService':add(DTLService,[]),
      'authHttpService':add(AuthHttpService,[
         'apiAuthService'
      ]),
      'authJsonHttpService':add(AuthJsonHttpService,[
         'apiAuthService'
      ]),
      'apiAuthService':add(ApiAuthService,['credentialManager']),
      'vaultService':add(VaultService,[
      ]),
      'server':add(HttpServerWrapper,['apiAuthService']),
      'dataManager':add(DataManager,['vaultService']),
      'credentialManager':add(CredentialManager,['dataManager'])
   }
};

function add(constructor,dependencies,optional_dependencies){
   return {
      "constructor":constructor,
      "dependencies":dependencies,
      "optional_dependencies":optional_dependencies
   }}
```

Тобто цей файл вміщує собі список всіх важливих модулів додатку та їх залежності від інших модулів додатку. Далі методом this.composeApplicationModules() додаток проходить по конфігурації модулів, будує модулі, що вже можливо побудувати і які не залежать від інших, запам’ятовує що ці модулі вже створені. В наступному такті побудови спробує створити модулі, які залежать від вже побудованих. Таким чином будівельник формує повністю робочий додаток з усіма модулями і їх залеєностями. Далі будівельник запускає об’єкт Application і припиняє працювати, адже його функцію виконано.

Об’єкт Application повинен мати можливість отримувати та надсилати HTTP-запити, тому він далі запускає модуль ApiBuilder, який в свою запускає модуль DtlModule. DtlModule запускає об’єкти DtlEndpoints і DtlHandlers, і таким чином сформує ендпоінти(записані в об’єкті DtlEndpoints), прив’яже їх до обробників запитів(в об’єкті DtlHandlers).

Основний вміст об’єкту DtlEndpoints – це масив з ключами-ендпоінтами та значеннями – необхідними обробниками.

```
endpoints(){
    return {
        [POST]:{
            'contract/create':'createContract',
            'contract/sign':'signContract'
        },
        [GET]:{
            'contract':'getAllContracts',
            'contract/:contractId':'getContract',
        },
        [DELETE]:{
        }
    }
}
```

В свою чергу в DtlHandlers містяться обробники:

```
handlersList() {
    return ['createContract','getAllContracts','getContract','signContract'];
  }
  createContract(data){
    return new Promise(async(resolve,reject)=>{
      try{
        let contractId=data['contractId'];//'CONTRACT03'
        let title=data['title'];//'Lease agreement';
        let description=data['description'];//'Car lease contract'
        let text=data['text'];//'Yadda yadda'
        let timestamp=Date.now().toString();
        let signatory1=data['signatory1'];//'Tetyana'
        let signatory2=data['signatory2'];//'Roman'
        let result = await
this.dtlService.createContract(contractId,title,description,text,timestamp,signatory1,sig
natory2);
        return resolve(result);
      }catch(e){
        return reject(e);
      }
    })
  }
  getAllContracts(data){
    return new Promise(async(resolve,reject)=>{
      try{
        let result = await this.dtlService.getAllContracts(data);
        return resolve(result);
      }catch(e){
        return reject(e);
```

```
    }
})    }
```

Далі ці дані передаються в об’єкт Server, який починає слухати HTTP-запити, які надходять на відповідні ендпоінти та реагує на згідно відповідним обробникам.

Обробники часто звертаються до методів об’єкту DtlService, який включає в себе логіку взаємодії з смарт-контрактом, блокчейном та іншими пов’язаними елементами.

Тому, з врахуванням всього вищеописаного, архітектура додатку виглядає як на рисунку 16:

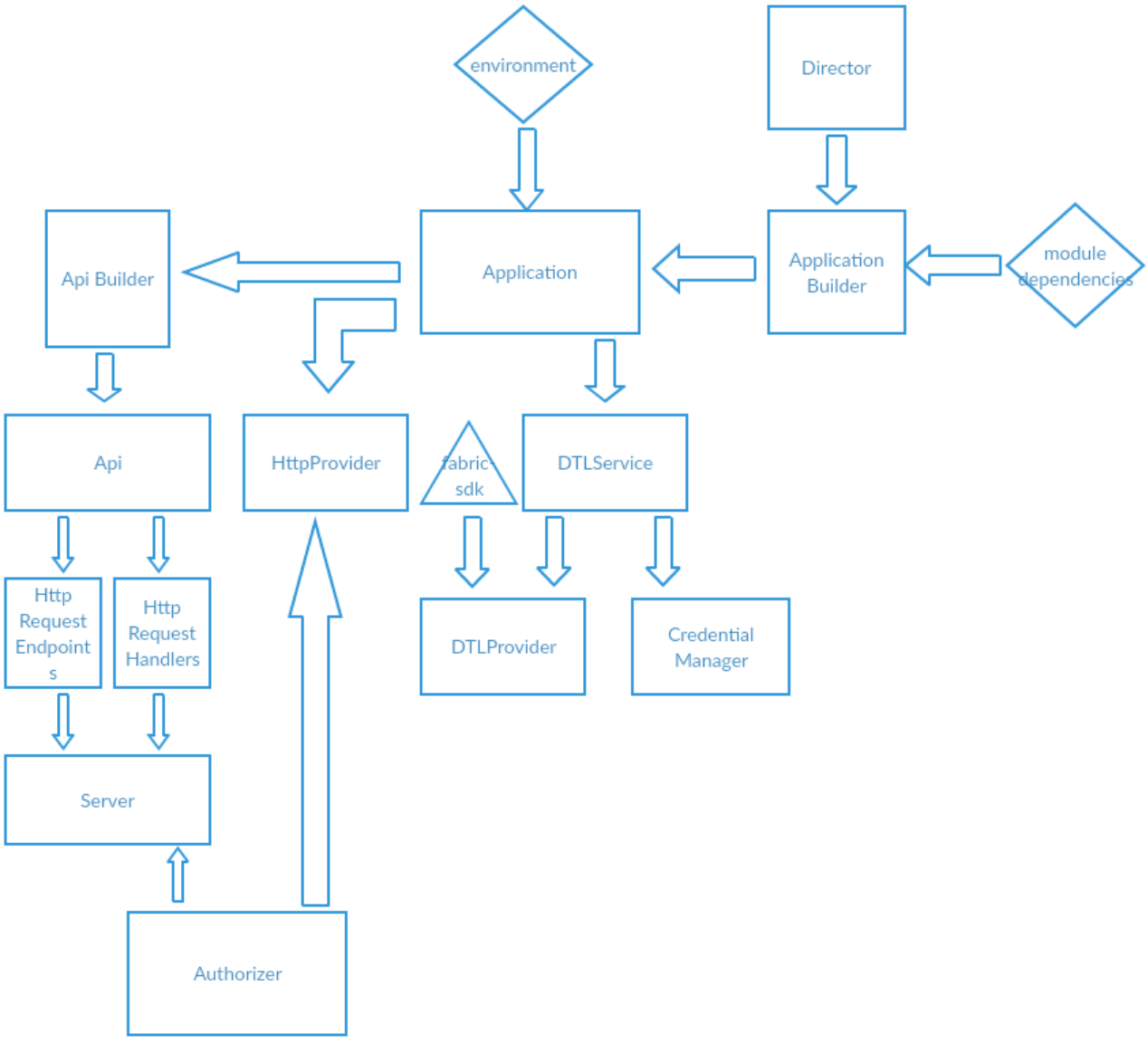

Рисунок 16. Архітектура додатку

6. Методика роботи розробника з інструментами розробки

Даний розділ надає інструкції щодо роботи розробників з інструментами розробки.

Загалом, алгоритм роботи включає наступні кроки:

- отримання доступу до ПЗ
- склонувати репозиторій на комп’ютер, де буде вестись розробка
- встановити програмне забезпечення Docker і Docker Compose
- зайти в директорію з скачаним репозиторієм DTLFabricFiles і запустити /dtl/startFabric.sh. Це створить 4 вузла(вони включають peer-додаток, копію couch-db) та ордерер і вмістить все в окремі Docker-контейнери. Також на кожен вузол буде встановлено смарт-контракт DTL. Далі цю мережу можна використовувати для тестування та розробки.

```
CONTAINER ID        IMAGE
                    NAMES
ddc1091f97ff        dev-peer1.org2.example.com-mycc-1.0-26c2ef32838554aac4f7ad6f100aca865e87959c9a126e86d764c8d01f8346ab
                    dev-peer1.org2.example.com-mycc-1.0
824bd5222052        dev-peer0.org2.example.com-mycc-1.0-15b571b3ce849066b7ec74497da3b27e54e0df1345daff3951b94245ce09c42b
                    dev-peer0.org2.example.com-mycc-1.0
9d130128c9e3        dev-peer1.org1.example.com-mycc-1.0-cd123150154e6bf2df7ce682e0b1bcbea40499416f37a6da3aae14c4eb51b08d
                    dev-peer1.org1.example.com-mycc-1.0
a057f3a8ec42        dev-peer0.org1.example.com-mycc-1.0-384f11f484b9302df90b453200cfb25174305fce8f53f4e94d45ee3b6cab0ce9
                    dev-peer0.org1.example.com-mycc-1.0
fe02713f487c        hyperledger/fabric-tools:latest
                    cli
50d3dd8aab1e        hyperledger/fabric-peer:latest
51/tcp                    peer0.org1.example.com
f786dba02811        hyperledger/fabric-peer:latest
51/tcp                    peer1.org1.example.com
e7bab7e7545d        hyperledger/fabric-peer:latest
51/tcp                    peer0.org2.example.com
b94c209f18fb        hyperledger/fabric-peer:latest
0051/tcp                  peer1.org2.example.com
accc93ba8320        hyperledger/fabric-couchdb
cp, 0.0.0.0:5984->5984/tcp   couchdb0
91d2216b0929        hyperledger/fabric-ca:latest
0:8054->8054/tcp          ca_peerOrg2
38daf0ac497f        hyperledger/fabric-couchdb
cp, 0.0.0.0:7984->5984/tcp   couchdb2
ebfa8b311017        hyperledger/fabric-orderer:latest
50/tcp                    orderer.example.com
eb8323442f71        hyperledger/fabric-couchdb
cp, 0.0.0.0:8984->5984/tcp   couchdb3
```

Рисунок 17. Консоль із запущеними контейнерами з елементами мережі

- встановити мову програмування NodeJs, пакетний менеджер NPM
- запустити команду встановлення необхідних пакетів та бібліотек npm install
- запустити серверний додаток API.

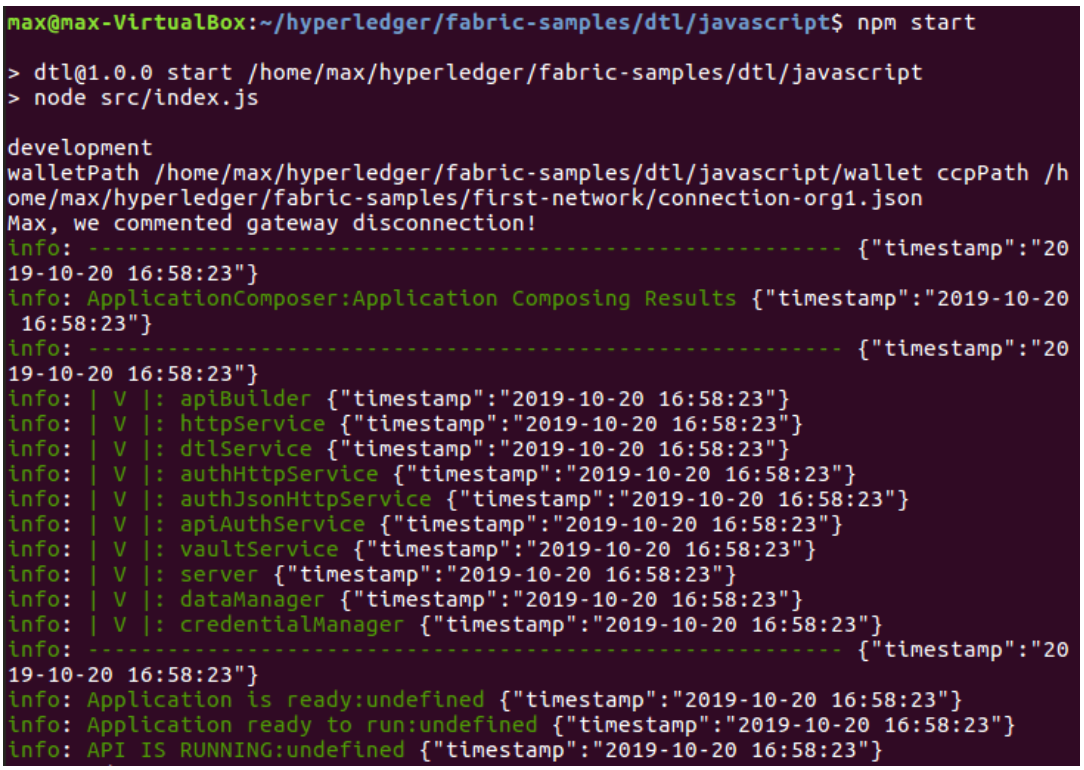

Рисунок 18. Консоль при запуску додатку

Для полегшення взаємодії з системою та прискорення процесу розробки додатків на основі запропонованих інструментів розробки, також існують скрипти що дозволяють одним запуском згенерувати криптографічні матеріали для створення та реєстрації в мережі користувача-адміна та користувача-клієнта.

Так, скрипт enrollAdmin.sh дозволяє зчитати конфігурацію мережі та тестові вхідні дані для генерації x509 сертифіката адміністратора, звертається до мережі для реєстрації цього адміна та зберігає його криптографічні дані в папці на файловій системі для подальшого користування. Не рекомендується використовувати цей скрипт в продакшн-оточенні, так як він створений виключно для прискорення процесу розробки та відладки.

Також використовується ще скрипт registerUser.js, який також дозволяє зчитати конфігурацію мережі та тестові вхідні дані для генерації x509 сертифіката користувача в межах конкретної організації, звертається до вузла для реєстрації цього користувача та зберігає його криптографічні дані в папці на файловій системі для подальшого користування. Як і enrollAdmin.js, не рекомендується використовувати цей скрипт в продакшн-оточенні, так як він створений виключно для прискорення процесу розробки та відладки.

7. Методика роботи користувача з програмною системою

Для того, щоб почати користуватись системою, потрібно виконати наступні кроки:

- Бути прийнятим як користувач в одній з організацій мережі. Для цього адміністратор мережі повинен включити ім’я та ідентифікацію користувача в конфігурацію організації та присвоїти йому певну роль та доступи в Службі управління членством(MSP).
- Отримати криптографічні матеріали цифрової особистості, що відповідає певному користувачу, та надійно зберегти їх в файловій системі комп’ютера, на якому так само розміщений вузол мережі. Ці матеріали можуть бути передані користувачу адміністратором.
- Створити HTTP запит на створення, підписання або зчитування договорів з смарт-контракту. Так як наразі існує багато способів створення HTTP запитів, для демонстрації було вирішено створити веб-інтерфейс для взаємодії з API. Даний веб-інтерфейс є сторінкою з HTML розміткою, підключеним javascript-кодом що дозволяє формувати HTTP запити на надсилати їх для обробки на стороні API, а також отримувати відповідь від додатку щодо результатів обробки запитів. Інтерфейс має 2 блоки – “Create Contract”(форма що дозволяє заповнювати дані договору та надсилати їх на API) та “Get Contract”(форма в якій можна переглянути дані договору та підписати його). Візуальну частину веб-інтерфейсу можна побачити на рисунку 19.

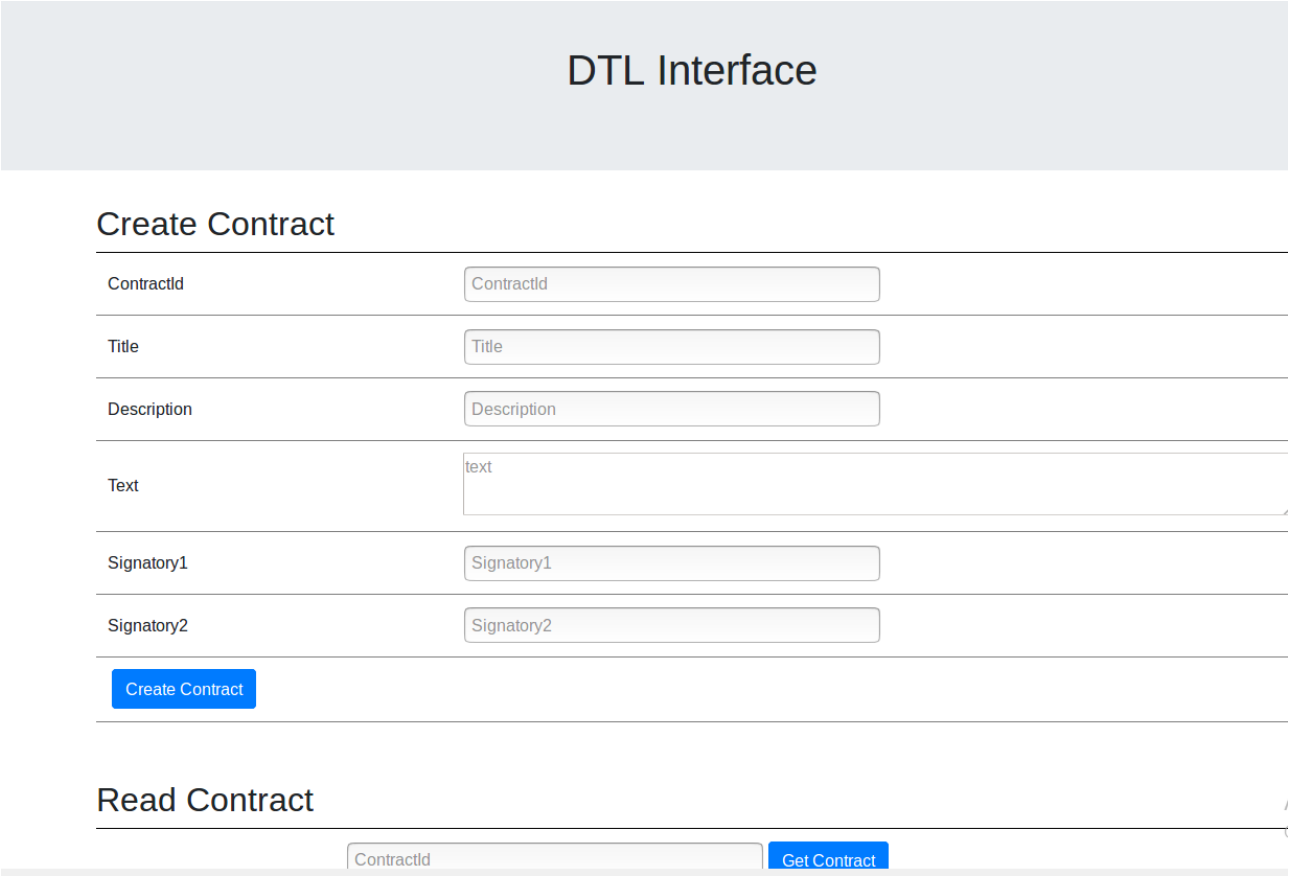

Рисунок 19. Веб-інтерфейс

Використовуючи даний інтерфейс, зручно заповнити поля його блоку “Create Contract”. Зразок заповнення полів інтерфейсу можна побачити на рисунку 20.

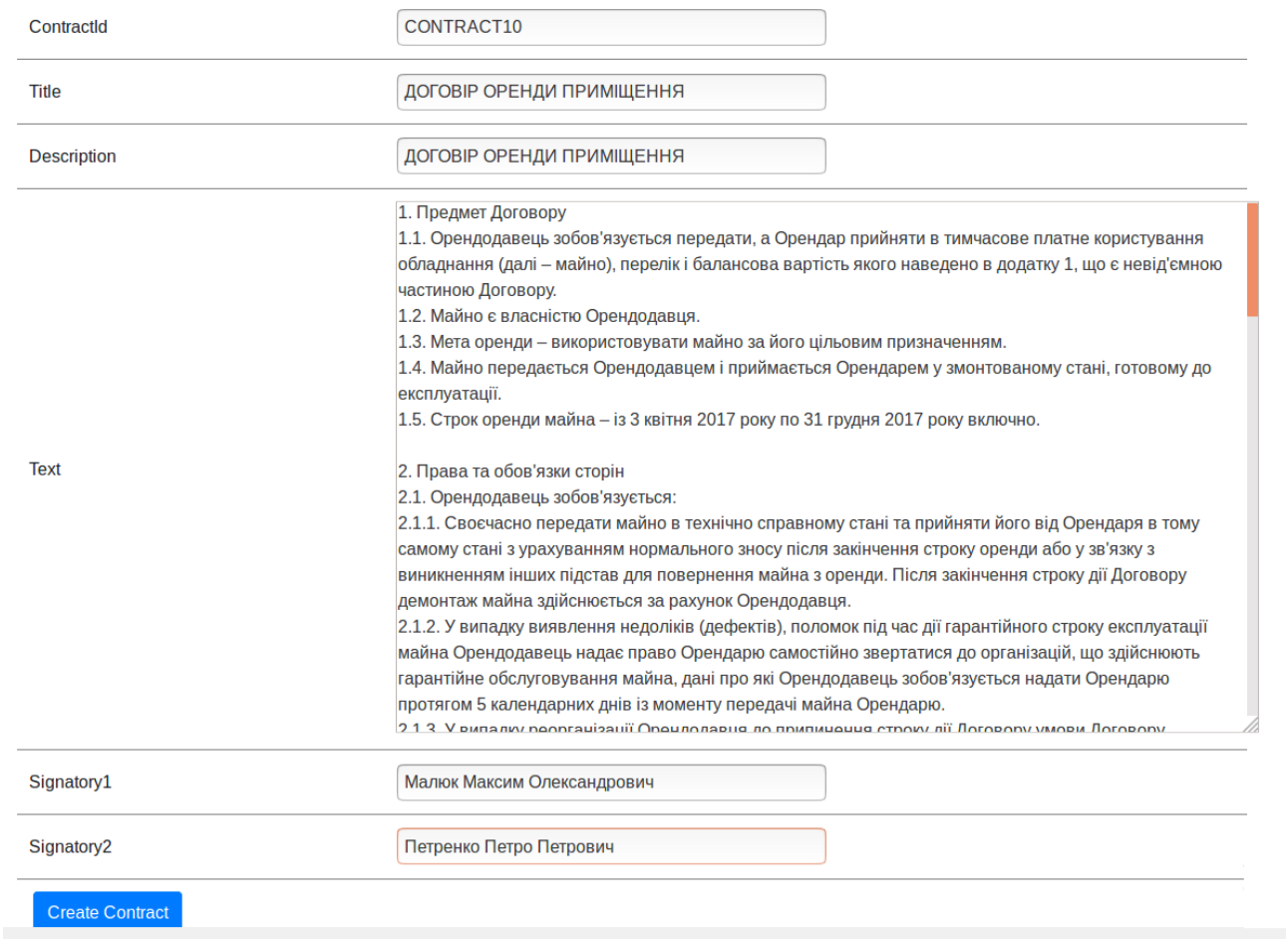

Рисунок 20. Зразок заповнення форми для створення договорів

Після заповнення всіх полів можна натиснути на кнопку “Create Contract”. Тоді інтерфейс сформує HTTP POST-запит на адресу API з даними, що відповідно були введені користувачем. Після нацей запит буде оброблено API, перероблено в запит, який буде сприйнято смарт-контрактом, що працює на Hyperledger Fabric.

Також веб-інтерфейс дозволяє зчитувати та підписувати договори, що збережені в децентралізованому реєстрі. Для виведення даних про договір необхідно ввести його унікальний ідентифікатор ContractId в спеціальне поле інтерфейсу і натиснути «Get Contract».

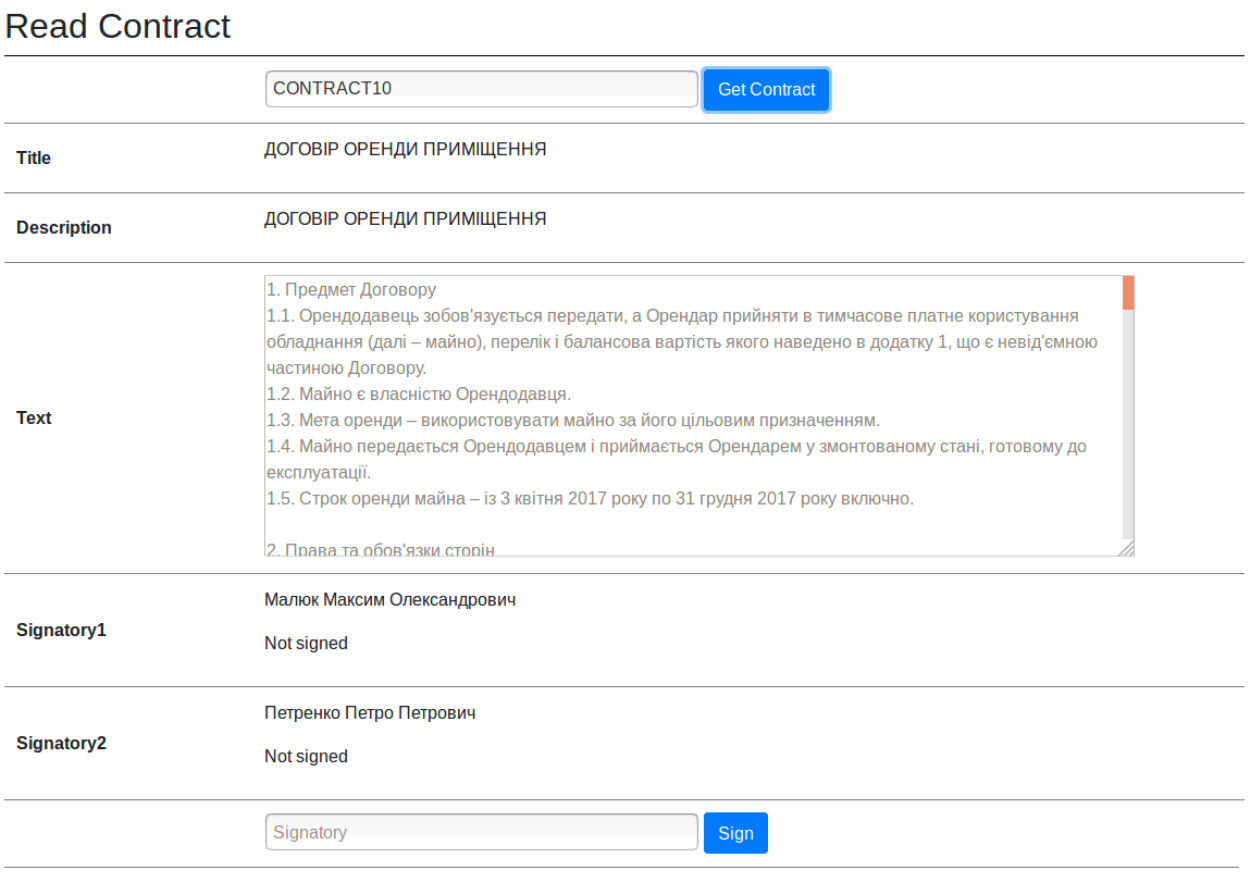

Рисунок 21. Зчитування даних договору та виведення їх в веб-інтерфейсі

Далі договір, виведений в інтерфейсі, можна підписати. Для цього достатньо ввести в поле Signatory(унизу сторінки) ім’я сторони, від імені якої проводиться підписання. Далі необхідно натиснути кнопку «Sign» і тоді інтерфейс надішле до API запит на підписання договору. В свою чергу API спершу порівняє чи відповідає ім’я підписанта імені користувача, приватний ключ якого використовується для підписання транзакції, яка буде сформована. Якщо дані співпадають, API формує запит до смарт-контракту на підписання договору, після чого у договорі залишається помітка що від був підписаний саме цим користувачем.

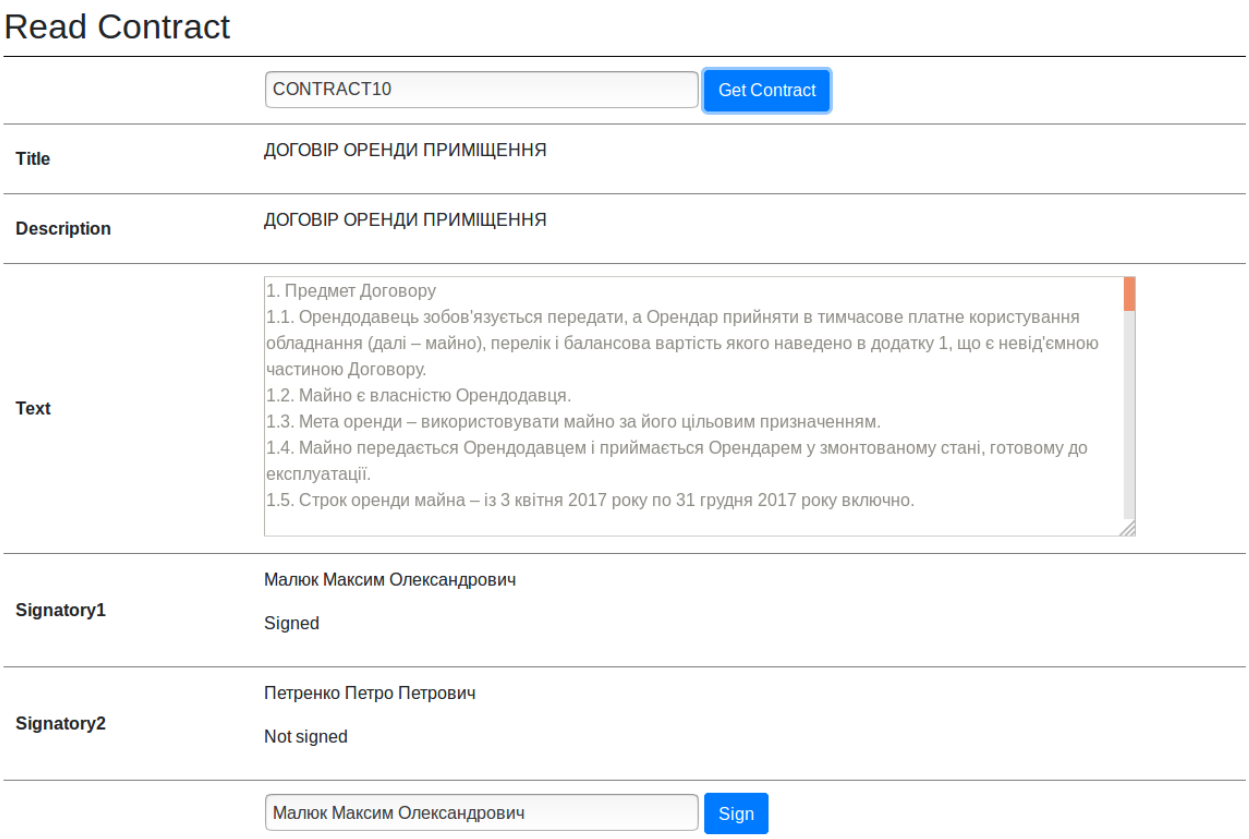

Рисунок 22. Договір з одним підписом

Відповідно інший користувач також може підписати договір, якщо у нього є приватний ключ, що зареєстрований в мережі на ім’я іншого підписанта, та здійснити таку ж операцію як в попередньому кроці.

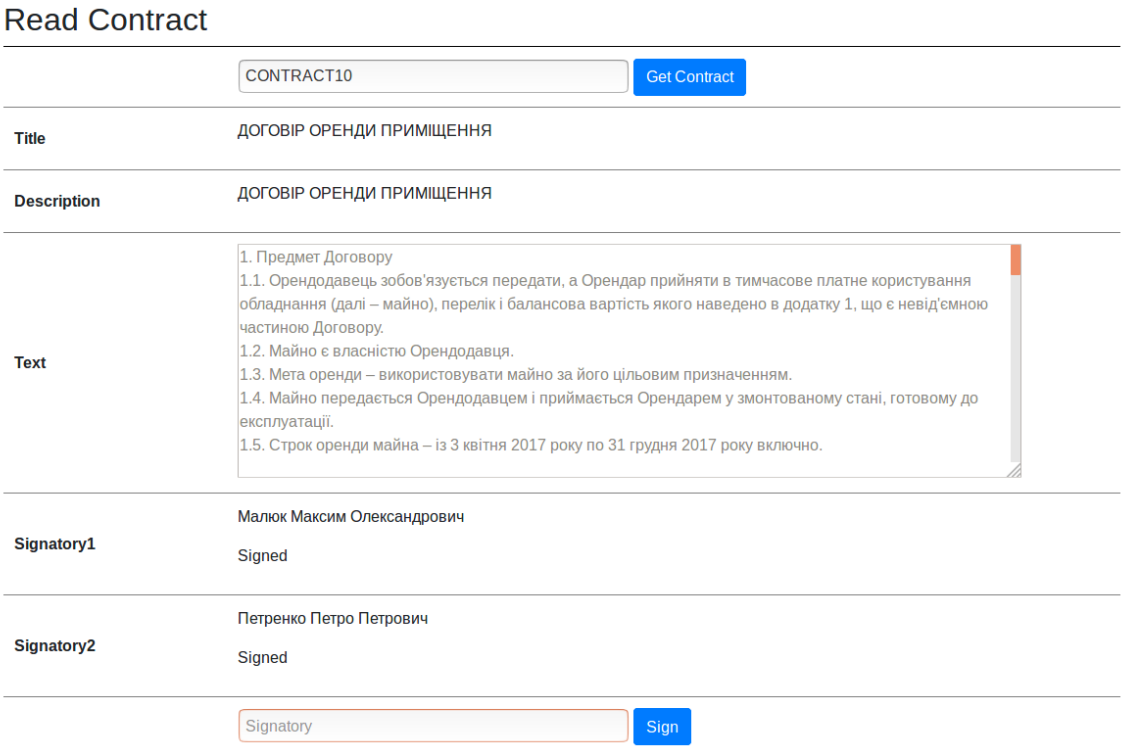

Рисунок 23. Договір з обома підписами

Результати виконання запитів на створення та зчитування договору, як воно відображено в логах API:

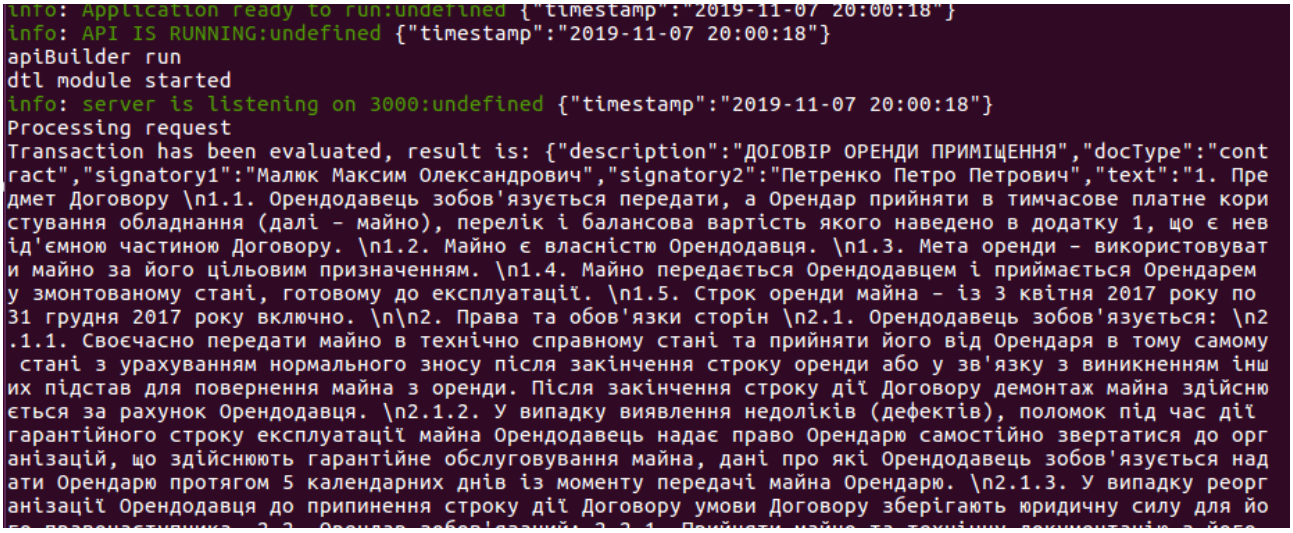

Рисунок 24. Логи в API, що виникають на запит створення контракту в децентралізованому реєстрі

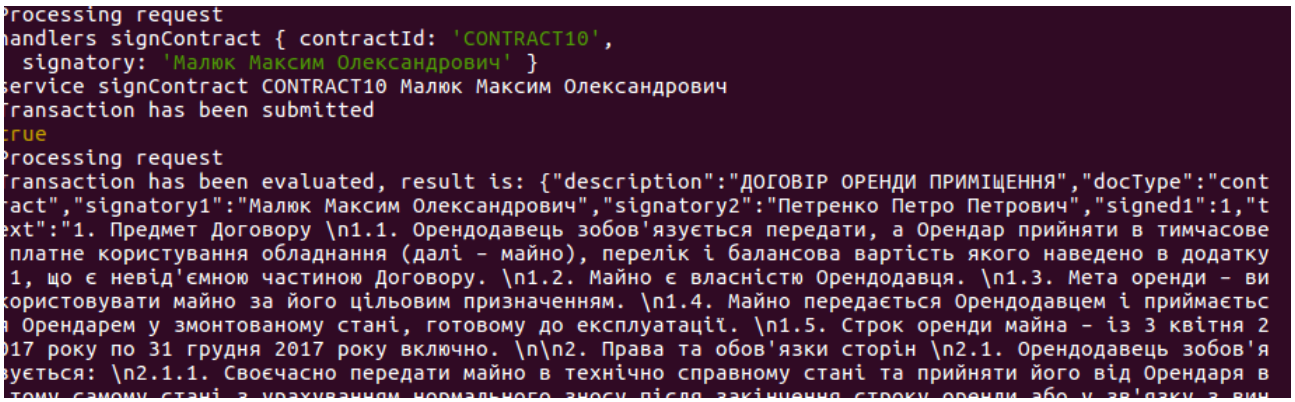

Рисунок 25. Логи в API, що виникають на запит підписання контракту в децентралізованому реєстрі

Отже, нам вдалося створити новий договір в реєстрі DTL, вивести про нього дані та підписати договір від імені обох сторін.

7. Стартап

Технологія децентралізованого реєстру дозволяє будувати на його основі різноманітні продукти. Так, наш децентралізований реєстр правочинів, що наразі надає можливість безпечно зберігати, змінювати та відтворювати договори в блокчейні, може органічно включити в себе ще функціонал з виконання та забезпечення умов цих договорів. Так, наприклад, якщо договори будуть зберігатись в форматі рікардійського контракту(одна частина заповнена читабельним для людини текстом, друга – виконуваним програмним кодом), то в реєстрі будуть міститися одночасно як умови угоди, так і програма, що виконує умови угоди. Наприклад уявимо звичайний договір підряду де за надання певних послуг однією стороною інша сторона зобов'язується зробити платіж на певні реквізити. Програма, що виконує зобов'язання другої сторони(платника) може сама перевірити стан та належність надання послуг і самостійно здійснити платіж на потрібні реквізити. У разі невиконання зобов'язань однією з сторін її до неї застосовуються запрограмовані в системі санкції. Таким чином, в одному продукті поєднується децентралізований реєстр договорів та інфраструктура для їх виконання та забезпечення.

Опис ідеї проекту (товару, послуги, технології)

Таблиця 1. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Продукт, що дозволяє децентралізовано та захищено зберігати зміст договорів, налає зручну систему для укладання та управління договорами та можливість	Публічні реєстри договорів в державному секторі	Відкритість та публічність даних для громадян та представників державних органів
	Приватні реєстри договорів для державного сектору	Захист конфіденційних даних, неможливість неавторизованого або

автоматичного та автономного виконання та забезпечення договорів системою	Приватні реєстри договорів для приватного сектору	прихованого втручання в зміст збережених документів, автоматичне виконання та забезпечення умов договорів

Таблиця 2. Аналіз потенційних техніко-економічних переваг ідеї

№ п/п	Техніко-економічні характеристики ідеї	Мій проект	Централізовані електронні реєстри	Паперові реєстри	W (слабка сторона)	N (нейтра-льна сторона)	S (сильна сторона)
1	Швидкість розгортання	W	S	S	+		
2	Дешевизна утримання	N	N	W		+	
3	Стійкість системи	S	W	W			+
4	Конфіденційність даних	N	S	W		+	
5	Відкритість даних	S	W	W			+
6	Захист цілісності даних	S	W	W			+
7	Автоматичне виконання договорів	S	N	W			+

8	Автоматичне забезпечення договорів	S	N	W			+

Технологічний аудит ідеї проекту

Таблиця 3. Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Децентралізований реєстр договорів	Hyperledger Fabric, спеціальна конфігурація мережі	З 2016 року	Технології розповсюджуються під MIT ліцензією
2	Back-office система управління договорами	Інтеграція з уже існуючими рішеннями з даним функціоналом(наприклад Gatekeeper, Icertis, ContractBook)	З 2006 року	Власна розробка або необхідна партнерська співпраця з компанією що володіє подібним продуктом
3	Синхронізація стану реєстру між усіма вузлами мережі	Постійна взаємодія вузлів з сусідніми вузлами щодо досягнення консенсусу	Kafka-protocol та Gossip-protocol активно використовується Hyperledger Fabric з 2016 року	Технології розповсюджуються під MIT ліцензією
4	Автоматичне	Віртуальна машина	Існує	Технологія у

	виконання та забезпечення грошових зобов’язань в договорах	для зчитування, виконання та забезпечення умов договорів програмним кодом з інтеграцією ескроу-агентів та платіжних систем		вільному доступі(віртуальна машина Bitcoin-script, Ethereum VM, EOS VM), API платіжних систем
5	Автоматичне виконання та забезпечення негрошових зобов’язань договорів	Віртуальна машина для зчитування, виконання та забезпечення умов договорів програмним кодом з інтеграцією що виконують зобов’язання, не пов’язані з передачею грошей	Для виконання немонетарних зобов’язань – нішова технологія або ще не існує	Потребує розробки

Аналіз ринкових можливостей запуску стартап-проекту

Останні кілька років не спостерігалась висока динаміка зростання інвестицій в лігал-тех компанії. Інвестори спостерігали за досить молодою діловою сферою і утримувались від великих угод. У 2016 році в галузь було вкладено 224 мільйони доларів; у 2017 році було вкладено 233 мільйони доларів. Як бачимо, поступово загальна кількість транзакцій зменшувалася, а середня сума однієї транзакції зростала. Але в 2018 році сталося вибухове зростання інтересу до індустрії, і загальна сума інвестицій стала більшою, ніж будь-коли раніше. В 2019 році станом на вересень публічно відомо про більш ніж 1.2 млрд інвестицій в цю ж сферу, що свідчить про посилення інтересу у розвитку сфери юридичних технологій.

Таблиця 4. Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	-
2	Загальний обсяг продаж, грн/ум.од	1.2 млрд
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відповідність нормам зберігання даних
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	10%

Основні фактори, які, як очікується, сприятимуть зростанню ринку програмного забезпечення для управління контрактами, включають зростання попиту на спритне програмне забезпечення для управління контрактами, посилення вимоги щодо відповідності систем правовим нормам та завдяки різноманітності моделей продаж та ліцензування.

Таблиця 5. Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що фор-мує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Контроль за тим як використовуються	Утримувачі приватних репозиторіїв	Усі сегменти ринку зацікавлені в безпеці умов	Дані повинні бути доступні лише

	дані	договорів, державні органи, приватні нотаріуси	домовленостей від неавторизованого доступу та недобросовісного використання персональних даних	авторизованим агентам, дані про запит на доступ повинен бути відомий усім зацікавленим особами що мають право на доступ до цієї інформації
2	Захист даних від неавторизованої зміни	Утримувачі приватних репозиторіїв договорів, державні органи, приватні нотаріуси	Всі сегменти ринку зацікавлені в безпеці умов домовленостей від неавторизованої зміни	Захист умов договору та вхідних даних від неавторизованої зміни є критичною необхідністю для впевненості сторін у передбачуваності результатів угоди після її виконання
3	Постійний та відкритий доступ до публічної інформації	Утримувачі приватних репозиторіїв договорів, державні органи,	Всі сегменти ринку зацікавлені в постійному та відкритому доступі до публічної	Постійність та відкритість доступу до інформації робить реєстр зручним засобом

		приватні та державні нотаріуси	інформації	електронного доказування
4	Автоматичне виконання та забезпечення договорів	Утримувачі приватних репозиторіїв договорів, державні органи	Всі користувачі, що користуються функцією рікардійських контрактів, зацікавлені в автоматичному виконанні контрактів без звернення до сторонніх органів(судів, арбітражів, виконавчих служб)	Автоматичне виконання та забезпечення умов договорів усуває потребу власноручного виконання та забезпечення умов договору та звернення до відповідних органів для забезпечення договорів у разі невиконання їх умов однією з сторін

Таблиця 6. Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Незацікавленість цільової аудиторії	Незацікавленість цільової аудиторії в імплементації нової системи в	Півот

		існуючі виробничі процеси	
2	Невизнання законодавцем	Електронні договори що самі виконують та забезпечують записані в них умови можуть не визнаватися законодавцем як автономні агенти або як альтернатива виконанню договору	Зміна юрисдикції або півот
3	Неконкурентноспроможність	Існуючий усталений конкурент входить на цільовий ринок та витісняє проект	Півот

Таблиця 7. Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Партнерська співпраця з існуючими	Існуючі постачальники системи	Надання можливості використовувати

	системами управління договорами	управління договорів не розглядають спосіб збереження договорів та їх автономність своїм ключовим функціоналом, тому наш проект можна інтегрувати з існуючими системами і використовувати суто для збереження та виконання договорів, в той час як партрерське ПЗ буде забезпечувати функціонал з управління договорами	функціонал продукту в інтеграції з існуючими системами управління договорами
2	Поширення технологій інтернету речей	Більше об'єктів реального світу стають підключені до	Розробка технологій смарт контрактів що впривають на

		мережі інтернет, що дозволяє впливати на поведінку цих предметів з цифрового світу	інтернет-речі та включення їх до віртуальних машин для виконання умов договорів, що не пов'язані з платежами

Таблиця 8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентос-проможною)
Тип конкуренції	Чиста конкуренція	Наразі в Україні немає публічних даних про конкурентів які би утримували значну частку ринку	Успішний маркетинг
Рівень боротьби	Рівень боротьби: Національний	Продукт орієнтований на Україну	Успішний маркетинг
За галузевою ознакою	Міжгалузева	Продукт може використовуватись в різних галузях	Успішний маркетинг

		економіки	
за видами товарів	товарно-видова	Усі потенційні продукти-конкуренти можуть бути одного типу: реєстри	Успішний маркетинг
За характером конкурентних переваг	Нецінова	Відмінність від існуючих конкурентів полягає окремих характеристиках продукту	Успішний маркетинг
За інтенсивністю	не марочна	Продукт не є подібним до жодного існуючого конкурентного продукту в Україні	Успішний маркетинг

Таблиця 9. Аналіз конкуренції в галузі за М.Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти
	Платформи, що дозволяють укладати та виконувати прості цивільні договори підряду в інтернеті(Upwork, Upcounsel)	Індивідуальні реєстри нотаріусів, реєстри закупівель, приватні реєстри договорів компаній в усіх галузях	Постачальники API для виконання та забезпечення видів умов договорів	Концентрація споживачів, вартість перемикання споживача, доступність інформації для

				споживачів є факторами
Висновки:	Укладені договори стосуються лише зобов'язань оплачуваного підряду, укладувані договори є однотипними, реєстр договорів є централізованим	Є можливість ввійти в ринок, конкуренти не мають аналогів для ключових якостей продукту(є централізованими або не-електронними реєстрами без можливості автоматичного виконання чи забезпечення)	Постачальники можуть обмежити або ускладнити користування програмними засобами, що є необхідними для включення функціоналу виконання та забезпечення певних видів зобов'язань договорів(наприклад, платіжні системи для виконання грошових переказів, постачальники ескроу-послуг, постачальники послуг в сфері інтернету речей)	Клієнти диктують умови якості, доступності та сприйняття продукту

Таблиця 10. Обгруднтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспро-	Обґрунтування (наведення чинників, що

	можності	роблять фактор для порівняння конкурентних проектів значущим)
1	Контроль за тим як використовуються дані	Захист персональних та конфіденційних даних від неправомірного доступу є ключою вимогою зберігання чутливих даних в реєстрах
2	Захист даних від неавторизованої зміни	Захист умов договору та вхідних даних від неавторизованої зміни є критичною необхідністю для впевненості сторін у передбачуваності результатів угоди після її виконання
3	Постійний та відкритий доступ до публічної інформації	Постійність та відкритість доступу до інформації робить реєстр зручним засобом електронного доказування
4	Автоматичне виконання та забезпечення договорів	Автоматичне виконання та забезпечення умов договорів усуває потребу власноручного

		виконання та забезпечення умов договору та звернення до відповідних органів для забезпечення договорів у разі невиконання їх умов однією з сторін

Таблиця 11. Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у по-рівнянні						
			-3	-2	-1	0	+1	+2	+3
1	Контроль за тим як використовуються дані	7				Нішові платформи укладення цивільних договорів	Індивідуальні реєстри компаній	Індивідуальні реєстри нотаріальних контор	
2	Захист даних від неавторизованої зміни	7					Нішові платформи укладення цивільних договорів, Індивідуальні реєстри нотаріальних контор, Індивідуальні реєстри нотаріальних контор		
3	Постійний та	12	Індивідуал			Індивідуал	Нішові		

	відкритий доступ до публічної інформації		ьні реєстри нотаріальн их контор			ьні реєстри компаній	платформи укладення цивільних договорів		
4	Автоматичне виконання та забезпечення договорів	16	Індивідуал ьні реєстри нотаріальн их контор, Індивідуал ьні реєстри нотаріальн их контор,			Нішові платформи укладення цивільних договорів			

Таблиця 12. SWOT- аналіз стартап-проекту

Сильні сторони:	Слабкі сторони:
1. Контроль за тим як використовуються дані 2. Захист даних від неавторизованої зміни 3. Постійний та відкритий доступ до публічної інформації 4. Автоматичне виконання та забезпечення договорів	1. Ціна перемикання 2. Низька обізнаність користувачів
Можливості:	Загрози:
1. Партнерська співпраця з існуючими системами управління договорами 2. Поширення технологій інтернету речей	1. Можлива незацікавленість цільової аудиторії

Таблиця 13. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтов-ний комплекс заходів) рин-кової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Використання продукту лише для зберігання документів(не лише договорів)	Спрощення продукту зменшує витрати на розробку та масштабує потенційні ринки та цільову аудиторію	1 рік
2	Використання продукту лише для зберігання договорів(без автоматичного виконання та забезпечення)	Спрощення продукту зменшує витрати на розробку	1 рік
3	Вихід на інший ринок	Зміна цільового ринку може полегшити конкурентне становище і збільшити потенціал росту	2 роки

Розроблення ринкової стратегії проекту

Таблиця 14. Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Організації та	Так	10%	-	Мають місце

	консорціуми що мають потребу в приватних реєстрах договорів				спроби інтеграції
2	Публічні органи що мають потребу в публічних реєстрах договорів	Наявні прояви позитивної ініціативи	3%	-	Обмежено волею держави та складністю входження в державний сектор
3	Сторони договорів що потребують впевненості у виконанні та забезпечені умов відповідних договорів	Мала частка	<1%	-	Обмежено інформованістю та готовністю споживачів

Таблиця 15. Визначення ключових переваг концепції потенційного товару

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
	Контроль за тим як використовуються дані	Маркетинг спрямований за інформування користувачів щодо важливості захисту персональних	Захист персональних та конфіденційних даних від неправомірного доступу є ключою вимогою	Безпека Конфіденційність Захист

		даних	зберігання чутливих даних в реєстрах	
	Захист даних від неавторизованої зміни	Маркетинг спрямований на пояснення ризику неавторизованої зміни даних	Захист умов договору та вхідних даних від неавторизованої зміни є критичною необхідністю для впевненості сторін у передбачуваності результатів угоди після її виконання	Непорушність Незмінність Захист
	Постійний та відкритий доступ до публічної інформації	Маркетинг спрямований на наголошення на постійності та відкритості публічних даних	Постійність та відкритість доступу до інформації робить реєстр зручним засобом електронного доказування	Відкритість Публічність Постійність
	Автоматичне виконання та забезпечення договорів	Макретинг спрямований на рекламування продукту як самостійного агента забезпечення зобов'язань та концепції «код як закон»	Автоматичне виконання та забезпечення умов договорів усуває потребу власноручного виконання та забезпечення умов договору та звернення до	Впевненість Виконання Забезпечення

			відповідних органів для забезпечення договорів у разі невиконання їх умов однією з сторін	

ВИСНОВОК

В процесі виконання роботи було розроблено інструментальні засоби для створення інформаційних систем, призначених для безпечного створення та збереження юридичних документів з використанням сучасних технологій децентралізованого зберігання даних.

Інструментальні засоби включають в себе розгортувач мережі, смарт-контракт, серверний додаток та інтерфейс. З використанням даних елементів, розробник може швидко розгорнути децентралізований реєстр договорів та налаштувати інфраструктуру для взаємодії з реєстром через веб-інтерфейс.

Системі, розроблюваній за допомогою наведених інструментальних засобів, характерні наступні характеристики:

- стабільність роботи навіть при високому навантаженні системи
- неможливість прихованої зміни критичних даних
- ефективність та надійність зберігання даних
- можливість збереження конфіденційності даних без порушення принципів публічності та відкритості для уповноважених учасників
- дешевизна зберігання даних.

Користувачами інструментальних засобів можуть бути розробники мереж, додатків і смарт-контрактів яким потрібно розробити децентралізовану систему для збереження договорів в децентралізованому сховищі. Також користувачами засобів є адміністратори, яким необхідно швидко інфраструктуру і мережу децентралізованого реєстру

Користувачами систем, що розробляються за допомогою даних інструментальних засобів, можуть бути фізичні та юридичні особи, що прагнуть безпечно зберігати підписані правочини, мати до них постійний гарантований доступ та водночас забезпечувати конфіденційність змісту даних договорів. Програмне забезпечення може бути впроваджено у міжособний документообіг, використане в наступних галузях: право, фінанси, банківська справа, державні закупівлі, цивільні та господарські правовідносини.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nicola L. What is an Electronic Contract? [Електронний ресурс] / Laver LLB Nicola – Режим доступу до ресурсу: http://www.inbrief.co.uk/contract-law/electronic-contracts/.

2. M. Gisler, K. Stanoevska-Slabeva, M. Greunz. Legal Aspects of Electronic Contracts Infrastructures for Dynamic Business-to-Business Service Outsourcing (IDSO'00). Stockholm, 5 - 6 June 2000 - P.1.

3. Операційна система [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0.

4. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System [Електронний ресурс] / Satoshi Nakamoto – Режим доступу до ресурсу: https://bitcoin.org/bitcoin.pdf.

4. NicholasJ. Szabo. Smart Contracts [Електронний ресурс] / NicholasJ. Szabo – Режим доступу до ресурсу: http://w-uh.com/download/WECSmartContracts.pdf.

5. DR. GAVIN WOOD. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDG [Електронний ресурс] / DR. GAVIN WOOD – Режим доступу до ресурсу: https://ethereum.github.io/yellowpaper/paper.pdf.

6. Diederick Cardon. Ricardian contracts — legally binding agreements on the blockchain [Електронний ресурс] / Diederick Cardon – Режим доступу до ресурсу: https://medium.com/ltonetwork/ricardian-contracts-legally-binding-agreements-on-the-blockchain-4c103f120707.

7. ConsenSys. Technical Introduction to Events and Logs in Ethereum [Електронний ресурс] / ConsenSys – Режим доступу до ресурсу: https://media.consensys.net/technical-introduction-to-events-and-logs-in-ethereum-a074d65dd61e.

8. web3.js - Ethereum JavaScript API Documentation [Електронний ресурс] – Режим доступу до ресурсу: http://web3js.readthedocs.io/en/1.0/.

9. NodeJs Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org.

10. Demiro Massessi. Public Vs Private Blockchain In A Nutshell [Електронний ресурс] / Demiro Massessi – Режим доступу до ресурсу: https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f.

11. Build Network - Hyperledger Fabric Tutorials [Електронний ресурс] – Режим доступу до ресурсу: https://hyperledger-fabric.readthedocs.io/en/release-1.2/build_network.html.

12. Smart Contract - Hyperledger Fabric Tutorials [Електронний ресурс] – Режим доступу до ресурсу: https://hyperledger-fabric.readthedocs.io/en/release-1.2/build_network.html.

13. Deep Dive - Hyperledger Fabric Tutorials [Електронний ресурс] – Режим доступу до ресурсу: https://hyperledger-fabric.readthedocs.io/en/release-1.2/build_network.html.

14.

Perfectial. How Companies Can Leverage Private Blockchains to Improve Efficiency and Streamline Processes [Електронний ресурс] / Perfectial – Режим доступу до ресурсу: https://perfectial.com/blog/leveraging-private-blockchains/.

15.Оголошення про продаж з першого повторного аукціону майна банкрута - ФОП Барна Ф.П. [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwiZs9-j1LXlAhXK4KYKHRtwB_wQFjAAegQIABAC&url=http%3A%2F%2Fold.minjust.gov.ua%2Ffile%2F31891.docx&usg=AOvVaw1wkXbZal59Iae5KMlf9Gxx.

16. Liquid. What's the difference between public and private blockchains? [Електронний ресурс] / Liquid – Режим доступу до ресурсу: https://blog.liquid.com/whats-the-difference-between-public-and-private-blockchains.

17. Global Trade Review. R3 and Commerzbank take “significant” step towards blockchain integration [Електронний ресурс] / Global Trade Review – Режим доступу до ресурсу: https://www.gtreview.com/news/fintech/r3-and-commerzbank-takes-significant-step-towards-blockchain-integration.

18. Binance Academy. Разъяснение Delegated Proof of Stake [Електронний ресурс] / Binance Academy – Режим доступу до ресурсу: https://www.binance.vision/ru/blockchain/delegated-proof-of-stake-explained.

19. Proof Of capacity [Електронний ресурс] / Investopedia – Режим доступу до ресурсу: https://www.investopedia.com/terms/p/proof-capacity-cryptocurrency.asp.

20. PoET 1.0 Specification [Електронний ресурс] – Режим доступу до ресурсу: https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html.

21. Binance Academy. Byzantine Fault Tolerance Explained [Електронний ресурс] / Binance Academy – Режим доступу до ресурсу: https://www.binance.vision/blockchain/byzantine-fault-tolerance-explained.

22. EOS Foundation. EOS Constitution [Електронний ресурс] / EOS Foundation – Режим доступу до ресурсу: https://github.com/EOSIO/eos/blob/5068823fbc8a8f7d29733309c0496438c339f7dc/constitution.md.

23. Рокатанський М. Разбираемся в основах Blockchain: Задача Византийских Генералов. Часть 1 [Електронний ресурс] / М. Рокатанський – Режим доступу до ресурсу: https://habr.com/ru/company/otus/blog/467053/.

24. Агибалова А.В. Задача о византийских генералах [Електронний ресурс] / Агибалова А.В., Зеленова А.И. – Режим доступу до ресурсу: https://bitly.su/kKuA.

25. Sathish Kumar. The Ultimate Guide to Consensus in Hyperledger Fabric [Електронний ресурс] / Sathish Kumar – Режим доступу до ресурсу: https://www.skcript.com/svr/consensus-hyperledger-fabric/.

ДОДАТОК А

Апробація

УКР.НТУУ “КПІ ім.І.Сікорського”.ТВ-з8110мп

Аркушів 5

2019

Міжнародний центр науки і досліджень
(м. Київ)

МАТЕРІАЛИ V МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

«*СУЧАСНА НАУКА: ПРОБЛЕМИ І ПЕРСПЕКТИВИ*»

29-30 ЖОВТНЯ 2019 РОКУ
(частина ІІ)

Київ
МЦНіД
2019

ЗМІСТ

Малюк М.О.,
НТУУ «КПІ» ім. В. Сікорського
Теплоенергетичний факультет

АПОСТИЛІЗАЦІЯ ДОКУМЕНТІВ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ БЛОКЧЕЙН

У даному документі розглянуто проблему процедури апостилізації документів в Україні. Проаналізовано поняття апостиля, його основні характеристики, вимоги для надання йому юридичної сили. Запропоновано альтернативний механізм створення та збереження апостилів за допомогою криптографічних інструментів і технології блокчейн. Проаналізовано основні інструменти для реалізації даного механізму, запропоновано порядок операцій, необхідний для створення, збереження та опублікування електронного апостиля документа.

Нотаріальні документи, рішення суду, документи з РАГСів, дипломи, атестати ВНЗ, потребують апостилізації для того, щоб мати відповідну юридичну силу закордоном[1]. Апостиль - це спеціальна форма засвідчення документів, призначених для пред'явлення в іноземні посольства, консульства, різні державні структури чи освітні установи. Метою проставлення апостиля є підтвердження справжності підпису особи, що засвідчила документ, а не змісту документа.

В Україні апостиль можна отримати в будь-якому відділенні РАГС не залежно від місця прописки чи проживання, а процедура апостилізації триває 1-2 тижні. Така тривалість технічно нескладної процедури зумовлена недосконалістю матеріального забезпечення державних органів.

Таким чином, прискорення проходження процедури апостилізації є актуальною проблемою в сучасній Україні. Міністерство юстиції України взяло курс на застосування нових технологій(QR-коди для електронних апостилів[2]) для її вирішення. Одним з таких технологічних рішень може стати використання криптографії і технології блокчейн для апостилізації документів.

Типовий блокчейн є децентралізований реєстром в якому всі записи є криптографічно захищеними та будь-які зміни в один запис призводять до каскадних змін в інших записах. Класичному блокчейну характерні наступні особливості:

- практична неможливість неправомірного(не-протокольного) внесення, зміни або видалення даних
- відкритість, доступність публічних даних та історії їх створення, внесення і зміни
- природна стійкість до традиційних кібер-атак[3]

Таким чином завантаження інформації в блокчейн практично гарантує її збережність та незмінність в подальшому, що є критичним для будь-якої бази документів, зокрема, апостилів.

Механізм створення апостиля за технологією блокчейн обов'язково використовує криптографічні функції хешування та шифрування.

Хешування(Hash функція) - це математична функція, здатна перетворити потік байтів у строку символів, унікальну для саме цієї комбінації байтів.

Таким чином документ в електронній формі(що є по-суті потоком байтів) можна пропустити крізь хеш-функцію і отримати унікальний для цього документу відбиток, який надалі можна використовувати для апостилізації.

Так як метою проставлення апостиля є підтвердження справжності підпису особи, що засвідчила документ, а не змісту документа, то хешування не призведе до втрати важливих даних апостиля про документ.

Для перевірки особи підписанта блокчейн-апостиля варто також використати інший криптографічний інструмент - асиметричне шифрування. Ключовими елементами інфраструктури асиметричного шифрування є приватний та публічний криптографічні ключі. Ці ключі математично пов'язані так, що повідомлення, зашифроване одним(наприклад, приватним) ключем може бути розшифровано лише відповідним йому публічним, і навпаки[4]. Так як приватний ключ лишається відомим лише одній особі-власнику, а публічний може бути вільно розголошений, використання даної інфраструктури дозволяє легко перевірити факт того, що певний документ був підписаний певною особою, просто спробувавши розшифрувати цей документ публічним ключем особи. Якщо розшифрування успішне – документ був підписаний приватним ключем саме цієї особи.

Таким чином, використовуючи усі перелічені інструменти, процедура апостилізації включає наступні етапи:

1. Хешування документу(щоб отримати відбиток оригінального тексту і скоротити його розмір до кількох сот символів)

2. Підписати хеш приватним ключем уповноваженої особи.
3. Завантажити в децентралізований реєстр блокчейн.

В процесі отриманий апостиль має наступні характеристики:

- його хеш відповідає оригіналу документу, і будь-які найменші дності легко виявляються
- легко перевірити факт підписання апостиля конкретною особою - якщо хеш розшифровується її публічним ключем, значить був зашифрований її приватним ключем
- легко перевірити факт існування апостиля на певний момент часу.

Отже, використання технології блокчейн без додаткових модифікацій дозволяє створювати та зберігати апостилі та забезпечувати їх незмінність, підтвердження особи підписанта та підтвердження існування документа і апостиля в конкретний час - всі характеристики, що вимагаються від апостиля.

Список використаної літератури

1. https://zakon.rada.gov.ua/laws/show/995_082
2. https://minjust.gov.ua/dep/ddr/poryadok-prostavlennya-apostilya
3. https://en.wikipedia.org/wiki/Blockchain
4. https://en.wikipedia.org/wiki/Public_key_infrastructure

Мыславец А.В., Зборщенко А.А.
студенты технического факультета
Кузьменко А.И.,
доцент, кандидат технических наук
Университет таможенного дела и финансов

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ВЗАИМОДЕЙСТВИЯ АВТОМОБИЛЬНОГО И ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Транспорт – это крупнейшая и важная отрасль народного хозяйства, огромная сфера приложения человеческого труда, широчайшая область использования новейших результатов науки и техники, гигантская динамическая система, где необходимо теснейшее взаимодействие частей и подразделений. В этой сложной отрасли народного хозяйства взаимодействуют различные виды магистрального транспорта, а так же городской и промышленный транспорт. Несмотря на административно-хозяйственную самостоятельность, все виды транспорта находятся в известной зависимости друг от друга и оказывают существенное взаимное влияние на процесс и результаты своей работы. Единство назначения всех видов транспорта, а так же тесная взаимозависимость между ними позволяет рассматривать их как единую транспортную систему государства, функционирование которой обусловлено определенными объективными закономерностями.

В настоящее время основная масса грузовых и пассажирских перевозок осуществляются с участием двух и более видов транспорта. Так, 80% грузов, перевозимых железнодорожным транспортом, зарождается и погашается на подъездных путях, т.е. на промышленном транспорте. Примерно 90% грузов, прибывающих в морские порты, передаются на железнодорожный транспорт. Около 50% грузов речного транспорта поступает также на железные дороги. Большая доля нефтегрузов передается с трубопроводов (через базы) на железнодорожный, морской, речной и автомобильный виды транспорта. Автомобильный транспорт практически взаимодействует со всеми видами транспорта. Особенно велик удельный вес участия автомобильного транспорта в пассажирских перевозках, осуществляемых всеми другими видами транспорта. Однако условия для взаимодействия различных видов транспорта нельзя признать оптимальными.

Повышение эффективности смешанных ж.д. – автомобильных перевозок являются несколько факторов:

- применение централизованной системы завоза и вывоза грузов автотранспортом крупных специализированных автохозяйств;
- контейнеризация и пакетирование перевозок;
- концентрация перегрузочных, складских и других грузовых операций на небольшом количестве хорошо оснащенных станций и контейнерных пунктов с созданием оптимальной сети транспортно-складских баз (терминалов), выполняющих распределительные функции (центры дистрибуции);